UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CIUDAD UNIVERSITARIA

PREPROCESADOR MATFOR

DESARROLLO DE UN PREPROCESADOR QUE INTEGRE EN FORTRAN PROPOSICIONES DE REPETICION Y SELECCION, ASI COMO FUNCIONES PARA SIMPLIFICAR EL USO DE VECTORES Y MATRICES

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

SERGIO ARTURO GARCIA CORONA

Director: Ing Raymundo Hugo Rangel Gutiérrez

México, D. F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

ANTECEDENTES

El lenguaje **FORtran** (siglas correspondientes a fórmula translator), es un lenguaje de programación diseñado para una amplia utilización en las áreas matemáticas, científicas y tecnológicas.

FORtran es uno de los más importantes lenguajes en el área de la informática, éste es importante por su efectiva aportación de elementos científicos que le permiten a una organización estructurar y sistematizar tanto sus objetivos, como sus actividades a desarrollar. Ya que **FORtran** es el primer lenguaje de programación aceptado como estandar en el campo de proceso de datos, un programa **FORtran** escrito para un procesador **FORtran** específico, puede aceptarlo otros procesadores fortran diferentes con un mínimo de cambios.

El lenguaje **FORtran** fue originalmente desarrollado para aplicaciones concernientes a la manipulación de datos numéricos y fue liberado para los usuarios por **IBM** en 1957. En el período de 1957 a 1966, el lenguaje evolucionó como un lenguaje de programación científico y fue estandarizado en 1966. Las primeras versiones fueron llamadas **FORTRAN, FORTRAN II** y **FORTRAN IV.** Cada nueva versión efectuó unos pocos cambios en las instrucciones básicas e incluyó características adicionales. En 1966 American National Standard (ANS) adoptó un estandar voluntario de **FORtran.** La Organización Internacional de Standares (ISO) también definió el **FORtran** estandar.

El lenguaje **FORtran** fue desarrollado para aplicaciones numéricas y el estandar de 1966 refleja a esta clase de usuarios. A través de los años, sin embargo, las aplicaciones de FORtran evolucionaron de una estricta aplicación orientada numéricamente, a aplicaciones más generales como proceso de datos.

Una nueva versión para un nuevo estandar de **FORtran** surgió en marzo de 1976. Una versión final del nuevo estandar de FORtran fue aprobada en septiembre de 1977 y se conoce como **FORTRAN 77.**

El nuevo estandar de **FORtran,** agrega características al FORtran previo de 1966, no vuelve obsoletos a los viejos programas de FORtran, pero incrementa los alcances del lenguaje en las siguientes áreas: Recursos de entrada/salida, declaración de datos, facilidad en subprogramas, construcciones previamente limitadas a valores enteros y a una diversidad de mejoras. Las características, normalmente conocidas como extensiones, tienen ampliados los alcances de FORtran, pero el pago a todo esto es que disminuye la portabilidad de programas.

Paralelamente al desarrollo del estandar de **FORtran**, se desarrollaron compiladores especiales para FORtran orientados a la enseñanza. Los más conocidos de éstos, fueron desarrollados en la Universidad de Waterloo (Waterloo, Ontario, Canadá), y denominados **"WATFOR y WATFIV"**. Un compilador similar para las grandes computadoras Control Data es **"MNF"**. Los compiladores **WATFOR** y **WATFIV** fueron diseñados para proporcionar excelentes mensajes de diagnóstico de error a los estudiantes, efectuar ejecuciones rápidas de los programas a los mismos y atenuar algunas características de la propensión al error de FORtran. El nuevo estandar **FORTRAN 77** adoptó las características de **WATFOR, WATFIV** y **MNF**, de manera que el FORtran de American National Standard está recomendado como la base para la programción FORtran, tanto para estudiantes, como para programadores profesionales.

El lenguaje de programación **PAScal** fue el primero en incorporar, en forma coherente, los conceptos de la programación estructurada definidos por Edsger Dijkstra y C.A.R. Hoare. **PAScal** fue desarrollado en Zurich por Niklaus Wirth, en Eidgenossische Technische Hochschule. PAScal se deriva del lenguaje **ALGol** 60, pero es más completo y fácil de usar. En la actualidad está ampliamente aceptado como un lenguaje útil que puede ser implementado con eficiencia y como una excelente herramienta de enseñanza.

El lenguaje **APL** (a programming language) fue desarrollado por Iverson en los principios de los 60's, se implementó como un lenguaje interactivo en 1967 y se convirtió en popular entre los ingenieros y matemáticos (quienes necesitaron un versátil "calculador de escritorio" como auxilio en sus trabajos) en los años 70's.

El conjunto de operadores de **APL** es más completo que el de otros lenguajes convencionales como PL/I y PAScal, incluye extensiones a operaciones escalares para manejar vectores y matrices teniendo implícito el manejo de ciclos de las estructuras de control. Este énfasis sobre el poder expresivo en los niveles de expresiones, es apropiado a lenguajes interactivos, dado que el uso de los lenguajes interactivos en el modo de "calculador de escritorio" está relacionado con la evaluación de expresiones. La variedad de operadores y expresiones de APL permiten diversos caminos o formas de lograr cálculos mejores que los que ofrecen otros lenguajes.

El mayor alcance está dirigido para que el ingenio del programador conduzca a una gran satisfacción, pero también puede pasar que los programas que desarrolle sean más difíciles de leer, depurar o mantener.

El área de trabajo es un mecanismo muy eficiente de APL para definir módulos, los cuales contienen nombres de subrutinas y conjunto de datos. Hay extensiones para APL, tales como "APL#PLUS y APLSV" diseñados especificamente para permitir el uso de aplicaciones en el procesamiento de información a gran escala para APL. APL no tiene explicitamente variables tipificadas o estructuras de bloques; sin embargo, tiene la declaración "GOTO" como su única forma de transferencia de control. Incluso, no hay una declaración "IF-THEN-ELSE" y el salto o transferencia condicional se hace por la implementación de un truco (saltar a la etiqueta cero, es interpretado como la salida de una subrutina y saltar a una etiqueta que no esta bien definida se interpreta como un "CONTINUE", el cual no tiene efecto). Esta situación difiere marcadamente de lo conocido por la comunidad de programadores. APL ha tenido una fuerte aceptación entre los programadores, debido a que consideran que las declaraciones de tipos, estructuras de control y bloques, pueden ser deshechadas en los lenguajes en line futuros.

CAPITULO I

"INTRODUCCION"

FORtran es uno de los primeros lenguajes de programación aceptados como estandar, por tal motivo existe una gran cantidad de bibliotecas implementadas en este lenguaje que varias empresas mexicanas siguen utilizando. El poder adquisitivo de las empresas no les permite actualizarse en el ramo computacional al ritmo que es requerido (les implicaría un costo excesivo), siguen trabajando con **FORtran** y es el lenguaje primordial. Aunque actualmente se hayan podido hacer de algunas **PC's** (PERSONAL COMPUTER), su información más usual se encuentra en **FORtran** y más tratándose de empresas de investigación o del ámbito científico, donde la cantidad de información no cabría en una PC; la tienen almacenada en discos y/o cintas magnéticas. Por estos motivos, nació la idea de implementar un software que ayudara a los programadores que laboran en empresas que tienen un procesador **FORtran** a hacer más flexible la elaboración de sus programas.

Se pensó en el diseño y desarrollo de un preprocesador que le simplificara al programador la elaboración de sus programas. Su programa fuente será pasado por el preprocesador y éste le proporcionará el programa objeto. El preprocesador convertirá el seudoprograma fuente, a un programa fuente, aceptable por el procesador **FORtran** disponible en la empresa, lo ejecutará (si el usuario lo desea) y le dará el programa objeto correspondiente.

A través del siguiente diagrama se ilustra la secuencia lógica anteriormente mencionada.

El nombre que se le dio a este preprocesador fue el de **"MATFOR"** (**FORtran matemático**) y el seudolenguaje que éste puede interpretar se le llama "**MATFOR**".

En el desarrollo de este trabajo se hablará de **FORtran** refiriéndose a **FORtran 77**, del "**Preprocesador MATFOR**" y de la **Programación MATFOR**", correspondiendo a lo anterior mencionado.

En los capítulos subsiguientes se verán: El diseño, desarrollo, pruebas, sintaxis para la programación, errores, ejemplos y ejecución del preprocesador.

"OBJETIVO"

El objetivo del presente trabajo es el de implementar las estructuras básicas de control de la programación estructurada, que el lenguaje **FORtran** tradicional no posee en forma natural. Debido en gran parte, a que **FORtran** es uno de los lenguajes de alto nivel más antiguos y, cuando se diseño, no estaba en voga la tendencia modular y de formato libre, que poseen los lenguajes más recientes (relativamente), tales como: "**PAScal**", "**ADA**", "**C**",etc., también tiene como objetivo el desarrollar primitivas que permitan un uso simplificado en el manejo de vectores y matrices.

CAPITULO II

"ASPECTOS GENERALES"

II.1 IDENTIFICACION DEL PROYECTO

MATFOR (**FORtran matemático**), es un preprocesador que extiende la capacidad del lenguaje de programación **FORtran**, permitiendo tener en un programa, las estructuras básicas de control de la programación estructurada, proporcionándole así al usuario una mayor productividad en la elaboración de los mismos. Además, proporciona un conjunto variado de funciones para la manipulación de operaciones entre vectores y matrices.

El preprocesador "**MATFOR**" tiene un diagnóstico de errores que resulta de gran ayuda durante la traducción de programas.

II.2 ENFOQUES

Los enfoques matemático y estructural son los que rigen a nuestro preprocesador, pero sin excluir absolutamente ninguno de los enfoques que el programador le puede dar a **FORtran** en sus programas. Si algún programa en **FORtran** se le da un enfoque matemático, es de esperarse que este enfoque se simplifique en líneas de código con programación "**MATFOR**".

Aunque la esencia de "**MATFOR**" es matemática y estructural, puede aceptar cualquier enfoque que el programador requiera dar a sus programas, procesando sólo aquellas partes que son extensiones y transcribiendo las demás tal y como esten.

II.3 PROCESADOR Y PREPROCESADOR

Existen en el ambiente de la computación dos definiciones de procesador, una como Hardware y otra como Software.

a) **PROCESADOR:** Procesador es un mecanismo de hardware que ejecuta instrucciones. Un ejemplo de procesador es el **CPU**.

b) **PROCESADOR:** La combinación de mecanismos que realiza la traducción del lenguaje de programación al de máquina se le llama procesador. Generalmente el procesador es referenciado como un compilador.

En esencia, un procesador se puede ver como una caja negra (Hardware o Software) a la cual se le proporciona una entrada y nos da una salida correspondiente.

Para nuestros fines, nombraremos procesador al que tiene como objetivo convertir el programa fuente a programa objeto.

PREPROCESADOR: Tomando la definición de procesador que nos interesa, definiremos al preprocesador como el software capaz de convertir cierto código a un código fuente, compatible con el procesador al cual se le antepone.

II.3.1 VENTAJAS Y DESVENTAJAS DEL USO DE UN PREPROCESADOR

Las ventajas de usar un preprocesador son:

a) Existen menos violaciones a las reglas de programación estructurada.

b) Mayor confiabilidad.

c) Facilidad de manejo.

d) Alto mantenimiento.

e) La codificación en seudocódigo sirve de documentación.

f) Se reducen los esfuerzos de programación.

Las desventajas primordiales de usar un preprocesador son:

a) Alto costo de software
   (se tiene que implementar o comprar).

b) Problemas al cambiar de equipo.

c) Pérdida de estandarización del lenguaje.

d) No portables los programas desarrollados.

e) Mayor tiempo en la obtención del código objeto.

II.4 MERCADO

El mercado para este preprocesador, serán todas aquellas organizaciones que tengan un procesador **FORtran**, y aunque la implementación del preprocesador esta hecha en un procesador **FORtran** especifico, con un mínimo de cambios sería suficiente para que lo aceptara cualquier otro procesador.

Este preprocesador está dirigido para aquel usuario que programe en **FORtran**, ya que con sus extensiones le proporcionará una mayor flexibilidad en su programación y le simplificará considerablemente líneas de código.

CAPITULO III

"ANALISIS CONCEPTUAL"

III.1 ANALISIS

Al llevarse a cabo el análisis del lenguaje de programación **"FORtran"** se observó que carece de las estructuras básicas de control en proposiciones de repetición y selección que le permitan una programación más sistemática, así como de primitivas (funciones integradas) que simplifiquen el uso de vectores y matrices a gran escala.

Dado que **FORtran** es un lenguaje muy común, se realizó un estudio sobre qué empresas o instituciones tendrían un procesador FORtran y utilizaran bibliotecas implementadas en él. Entre las empresas e instituciones que se visitaron están:

a) INSTITUTO MEXICANO DEL PETROLEO (I.M.P.)

b) FACULTAD DE INGENIERIA DE LA U.N.A.M.

c) INDUSTRIAL MINERA MEXICO

d) COMISION FEDERAL DE ELECTRICIDAD (C.F.E.)

e) CENTRO DE INVESTIGACION DE ESTUDIOS AVANZADOS DEL I.P.N.

f) CENTRO DE PROCESAMIENTO DE DATOS ARTURO ROSENBLUETH

Se observó que un porcentaje considerable de programadores de cada empresa programan en **FORtran** y tienen una gran variedad de bibliotecas que se usan para fines específicos. Se mencionan algunas de las bibliotecas más comunes existentesen el mercado.

BIBLIOTECA	DESCRIPCION
Functional Mathematical Programming System **FMPS**	Paquete matemático, incluye modo de programación lineal
STAT PACK	Paquete estadistico, estadística descriptiva, estadística elemental, intervalos confidenciales, análisis de varianza, análisis de regresión y funciones de distribución.
International Mathematical and Statistical Libraries **IMSL**	Paquete estadistico y matemático. Análisis de datos experimentales, estadística básica, análisis de datos categóricos, ecuaciones diferenciales, generación y prueba de nums. random, etc.
CALCOMP	Paquete de graficación.
Scientific subroutine package **SSP**	Paquete matemático y estadístico.
LINPACK	Solución de sistemas de ecuaciones lineales y problemas relacionados.
ELLPACK 77	Solución de ecuaciones parciales elipticas.
MATH-PACK	Paquete matemático. Interpolación, integración numérica, solución de ecuaciones, diferenciación, polinomios, etc.
FORSYT	Paquete matemático.
LINWOOD	Paquete matemático.
STAT1	Análisis estadistico para un conjunto de observaciones, - análisis para una sola variable.
GRAFICA COM	Graficación de funciones tabulares o por ecuación dada por el usuario.

BIBGRAF COM	Graficación estadística, circular, histogramas, polinomios de frecuencia, diagrama de barras y poligonos.
MATRICES2 EXE	Algebra matricial, inversiones, triangularizaciones, simetrizaciones, sol. de sistemas de ecuaciones.
GEIDE EXE	Solución de sistemas de ecuaciones por el método de GAUSS -SEIDEL.
LPMOGG EXE	Método de optimización lineal por el método de la gran "M".
REGREMUL EXE	Regresión lineal múltiple para un máximo de 10 variables.
ARES EXE	Análisis de armaduras en el espacio.
ARPLA EXE	Análisis de armaduras en el plano.
MALLAPLANA EXE	Análisis de reticula plana.
MARES EXE	Análisis de marcos en el espacio.
MARPLA EXE	Análisis de marcos planos.
CIPW EXE	Cálculo de los minerales normativos para una muestra de roca ignea.
ROCKY2 EXE	Análisis de fractura de roca para la determinación de la orientación de planos.
LOGO EXE	Cálculos topográficos, áreas poligonales, curvas horizontales, verticales, etc.
CSMP EXE	Programa para generar la simulación analógica digital de un sistema dinámico por medio de su ecuación diferencial.

Como a la mayoría de las empresas el cambiar de procesador y de las bibliotecas correspondientes les implicaría un costo considerable, es por eso que se sigue manteniendo **FORtran**, aunado a los buenos resultados que presenta.

FORtran es uno de los lenguajes que ha venido evolucionando. La primera versión fue llamada **FORtran** en **1957**; después siguieron las versiones **FORtran II** y **FORtran IV**, en los años **1957** a **1966**, cuando American National Standard (**ANS**) adoptó un estandar de **FORtran**. Fue entónces hasta los mediados de los 60's, que se estandarizó FORtran. Cada versión efectuó algunos cambios en las instrucciones básicas e incluyó características adicionales. Pasaron 10 años para que surgiera un nuevo estandar de **FORtran** llamado **FORtran 77**. Este estandar agrega características al FORtran previo de 1966, aclara ciertas ambigüedades y efectúa algunos cambios.

Se pensó entonces en desarrollar un preprocesador que integrara ciertas características a **FORtran 77**, de las cuales carece, y permitirle así al programador una mayor productividad en el desarrollo de sus programas.

Se recomendó el uso de lenguajes que obligaran a una programación, basada exclusivamente en las estructuras básicas, (secuencia, condición, iteración,etc.), evitando así la tentación de usar instrucciones que dán lugar a estructuras complicadas, (como el **GO TO**, gran cantidad de **ciclos** explícitos).

Realizando una evaluación de los lenguajes de programación más afines al tema, fueron seleccionados el **PABcal** y el **APL**, ya que estos nos brindan las herramientas necesarias para implementárlas en **MATFOR** y satisfacer nuestro propósito.

Del lenguaje **PABcal** se obtuvieron las características de las estructuras básicas de control (repetición y selección), para ser implementadas en nuestro preprocesador. Las proposiciones elegidas fueron:

A) WHILE
B) REPEAT
C) FOR
D) CASE

Con respecto al lenguaje **APL**, de él se extrajo la idea del manejo de vectores y matrices, para lo cual se desarrollarán un conjunto de primitivas (funciones integradas) que nos permitirán un manejo más simplificado entre ellos.(Así como el uso de tablas para ganar velocidad en la respuesta).

Entre las primitivas a implementar estarán las siguientes:

A) LECTURA DE UN VECTOR
B) ESCRITURA DE UNA MATRIZ
C) MAXIMO DE UN VECTOR
D) MULTIPLICACION DE UN ESCALAR X MATRIZ
E) SUMA DE VECTORES
F) INVERSA DE UNA MATRIZ
G) ETC,ETC.

Es conveniente hacer hincapié en que la programación estructurada no consiste en evitar el uso de **GO TO's** , sino de que nuestro módulo al ser construido no sea pensado en el GO TO, puesto que no es realmente necesario desde el punto de vista de lógica, aunque ya en el momento de codificar el módulo bien estructurado, pueda hacerse uso de GO TO's , pero esto no disminuirá la estructuración lógica del módulo.

Para lograr el desarrollo de nuestro preprocesador se tendrá que crear un seudocódigo o sea, un lenguaje de mayor nivel en el cual sólo existan postulados que reflejen las estructuras básicas y las funciones intrínsecas de vectores y matrices que tendrá el preprocesador, se irá transformando esta codificación en otra equivalente, hasta llegar a la codificación del lenguaje FORtran.

Cabe mencionar que existen en el mercado algunos preprocesadores que transforman el lenguaje estructurado (seudocódigo) en un lenguaje normal (no estructurado), **RATFOR** es uno de ellos.

III.2 JUSTIFICACION

FORtran en ningún momento ha sido olvidado por las grandes organizaciones y universidades. Se sigue pensando en él como una herramienta de solución a problemas, así como en un lenguaje que, de acuerdo a la época, necesita ciertos cambios para que satisfaga ciertas o más necesidades de los usuarios, tal como lo pensó la Universidad de Waterloo en los años 60's, con el desarrollo del compilador **"WATFOR"**, creado con el fin de satisfacer requerimientos en la educación e investigación. Se pensó en la importancia de tener un compilador rápido que manejara una población estudiantil creciente y de manera económica. También en la necesidad de proveer diagnósticos, lo más completos posibles, con el próposito de dar independencia al estudiante para depurar sus programas.

En el área de investigación mucho del tiempo de cómputo es consumido en el desarrollo de programas antes de que se libere a producción. **WATFOR** acelera este proceso empleando un método de compilación rápida, con el cual resulta obviamente un mejor tiempo de respuesta; más aun, con el diagnóstico de errores, es de gran ayuda durante la programación. Posteriormente esta Universidad desarrolló el compilador **"WATFIV"** para dar capacidades adicionales a **"WATFOR"**.

Actualmente siguen pensando en **FORtran**, compañias tan importantes como **UNISYS** (Univac-Bourroughs), IBM, CONTROL DATA, DIGITAL, etc. (las grandes compañias que forman la ANSI). Existe un comité que se está reuniendo para tratar sobre los cambios e implementaciones que necesita FORtran, no se puede pensar en dejarlo tal y como está, y observar como otros lenguajes se hacen más poderosos que él. Estos cambios de los que se habla podrian salir en un nuevo estandar de FORtran, sólo hay que esperar (pienso que no mucho) que se pongan todos de acuerdo para establecer la norma y el FORtran de los 80's aparecerá.

UNISYS, a sus usuarios, les distribuyó el articulo que sacó uno de sus colaboradores, de la reunión tenida en abril de 1988, en Baltimore, Maryland; a dicho articulo lo llamó: "8X : FUTURE OF FORTRAN". Este documento trae todas las implementaciones que le pretenden hacer a FORtran (El autor del articulo es Rolison Lawrence y se encuentra en la bibliografia).

Entre las caracteristicas del FORtran 8x están:

- Programa fuente de formato libre.
- Mejoras en las declaraciones.
- Declaraciones y caracteristicas nuevas.
  - Nuevas declaraciones.
  - Operaciones de arreglos.
  - Tipos definidos por el usuario.
  - Módulos.
  - Procedimientos.
  - Control de presición.
- Viejos amigos estandarizados: Namelist.
- Mecanismos para borrar caracteristicas en desuso.

En el **anexo C** se encuentran algunas declaraciones de instrucciones para el FORtran de fines de los 80's.

El preprocesador **MATFOR** no está tan fuera de la realidad, ya que algunas de las caracteristicas que se presentan en el nuevo **FORtran**, las tendrá contempladas en él; además de otras ventajas que presentará **MATFOR** y que no presenta **FORtran 8X**, como son las primitivas para el uso a gran escala de vectores y matrices. Por estas razones pienso que no se está dejando, ni se dejará morir a **FORtran**, ya que es de esperarse que evolucione superando con ello sus limitaciones anteriores

III.3 EXTENSIONES

Con **MATFOR**, se extenderá a **FORtran**, en dos importantes aspectos:

a) Uso de las estructuras básicas de control que permitirán una programacion más sistemática.

b) Uso de funciones que le permitirán el manejo de vectores y matrices a un nivel mayor y simplificado.

III.4 ESPECIFICACIONES FUNCIONALES

MATFOR le simplificará al programador un gran número de líneas de código en la elaboración de sus programas. Le permitirá hacer uso de proposiciones especiales que le darán a sus programas mayor visibilidad y fácil comprensión para otros programadores. La capacidad de manejo de arreglos que tendrá el preprocesador, eliminará la necesidad de tener en forma explicita muchos ciclos que se requerirían en **FORtran**.

El preprocesador "**MATFOR**" detectará la mayoria de los posibles errores que pudiera cometer el programador al hacer su programa en el **seudocódigo MATFOR**, el diagnóstico de errores que tendrá proporcionará independencia al programador en la corrección de sus programas. Cada vez que el preprocesador se encuentre con una mala instrucción, desplegará un mensaje de error y parará el proceso, permitiéndole así corregir esa instrucción y ejecutar de nuevo.

III.5 LIMITACIONES

* El preprocesador puede llegar a trasladar una línea de código que se encuentre mal (puesto que no lleva a cabo un análisis semántico), pero el compilador FORtran está perfectamente capacitado para detectar cualquier error de sintaxis que pudiera pasársele al preprocesador

* Es una condición necesaria y suficiente para poder hacer uso de las primitivas de **MATFOR**, declarar las variables y arreglos a usar con la sintaxis que se presenta en la sección I del capitulo V.

* Es necesario colocar el caracter de reconocimiento de función (!) entre las columnas 1-5 para poder usar una primitiva de **MATFOR**, ya que de esta manera reconoce la función y si no tiene dicho signo transcribirá la línea tal y como esté.

Por ejemplo :

```
12345678901234567890.....
c Declarada previamente la variable "a" de tipo vector
c se usa una primitiva para leer sus elementos  y otra
c para escribirlos
c
  !   ?a
   !  &a
```

* No debe de utilizar el programador nombres de variables iguales a las palabras reservadas del preprocesador. Dichas palabras son:

a) WHILE	a.1) SUMVEC	a.2) SUMMAT
b) REPEAT	b.1) RESVEC	b.2) RESMAT
c) UNTIL	c.1) MULVEC	c.2) MULMAT
d) UNT	d.1) MVMVEC	d.2) MMVMAT
e) CASE	e.1) MVEVEC	e.2) MMEMAT
f) FOR	f.1) MEVVEC	f.2) MEMMAT
g) VAR	g.1) MAXVEC	g.2) INVMAT
h) ARRAY	h.1) MINVEC	h.2) IMPMAT
i) BEGIN	i.1) MEDVEC	i.2) LEEMAT
j) ENDW	j.1) IMPVEC	
k) ENDF	k.1) IMVVEC	
l) ENDC	l.1) LEEVEC	
m) ENDV	m.1) LEVVEC	
n) ENDA		
o) ID$1		
p) ID$2		
q) CHAR		
r) ENDB		

* Máximo número de dígitos para dimensiones es 10 y sólo acepta vectores y matrices de tipo entero, real y caracter.

* Etiquetas de la 70000 a la 70030 y de la 80000 a la 80500 no usarlas, son internas del preprocesador.

* Las declaraciones para variables de tipo **complex** y **double precision** deben de ser simples. En el caso que se requieran de otro grado, declararlas fuera del grupo de variables encerradas por **VAR--ENDV** o **ARRAY--ENDA**

Por ejemplo :

```
      COMPLEX A,BB,CCC
      DOUBLE PRECISION UNO,DOS
C  FORMA SIMPLE
C
      COMPLEX C1,C2*16,C3
      DOUBLE PRECISION ARRAY(15),LIST2(2,100)
C  FORMA NO-SIMPLE
```

III.6 REQUERIMIENTOS DE SOFTWARE Y HARDWARE

SOFTWARE : Los programas se desarrollarán en :

a) Lenguaje de programación **FORtran ascii** level 11R1

b) Lenguaje **Ensamblador ASM** level 4R1A

El sistema operativo a utilizar sera **EXEC-8** level 39R3D

HARDWARE :

a) Una computadora **UNIVAC 1100/80**

b) Pantalla de video modelo **UTS-20** (Universal Terminal Sperry)

c) Almacenamiento en disco magnetico (70 trakc's)

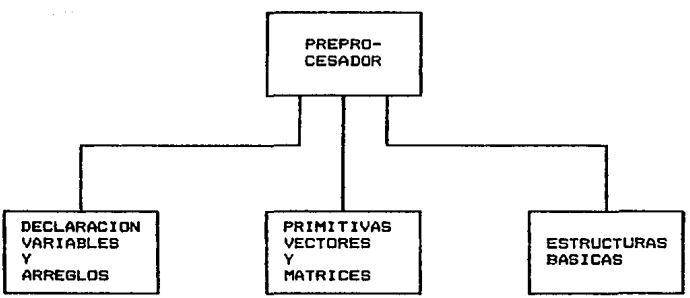
d) Impresora Unisys (2000 LPM)

En el caso que se requiera la instalación del preprocesador en otro equipo, será necesario hacer un mínimo de cambios en los programas en **FORtran** y la conversión de los programas en ensamblador.

III.7 ELEMENTOS QUE INTEGRARAN AL PREPROCESADOR

El preprocesador **"MATFOR"** constará de 3 módulos con los cuales se garantizarán las características previamente mencionadas que MATFOR agregará a **FORtran**. Dichos módulos son: declaración de variables y arreglos, primitivas para vectores y matrices y estructuras básicas.

REPRESENTACION GRAFICA



DECLARACION DE VARIABLES Y ARREGLOS: Este módulo le permitirá al programador declarar sus variables y arreglos de una manera más flexible y facil de entender.

PRIMITIVAS PARA VECTORES Y MATRICES: Aqui se definirá el conjunto de primitivas que nos facilitarán la manipulación de vectores y matrices a gran escala. La sintaxis para utilizar estas funciones será de una manera autodocumental para lograr que se recuerden facilmente y hacer más ágil la programación.

ESTRUCTURAS BASICAS: Módulo el cual tendrá contemplada la conversión de las estructuras básicas (repetición y selección) a su equivalecia a FORtran. Las proposiciones de repetición y selección serán parecidas a las de PAScal.

CAPITULO IV

"D I S E Ñ O"

IV.1 DESCRIPCION Y DIAGRAMA GENERAL DEL PREPROCESADOR

PROGRAMA PRINCIPAL: El programa principal se encarga de llamar a las subrutinas en ensamblador para ser llamado el programa como preprocesador y a las subrutinas para inicializar tablas. Verifica si la opción mandada en el preprocesador es correcta y si los nombres de los archivos mandados como parámetros existen o no, sobre el sistema. En este programa se realiza la verificación de la programación MATfor , examina si existen proposiciones y funciones que debe traducir y las lleva a cabo.

El preprocesador está integrado por las siguientes subrutinas:

SUBRUTINA ARCHIV: En esta subrutina, se genera la identificación del preprocesador y se obtienen los parametros mandados en él (archivo de entrada y archivo de salida) a través de subrutinas en ensamblador.

SUBRUTINA TABINI: Aqui se llevan a cabo las inicializaciones. Se inicializa la tabla de variables y la tabla de estatus donde se va registrando el llamado a primitivas de vectores y matrices.

SUBRUTINA OPT: Subrutina en ensamblador creada con el fin de pasarle a FORtran en una palabra, las opciones con las cuales fue ejecutado el preprocesador.

SUBRUTINA VAR: En esta parte se realiza la conversión de la declaración de variables y arreglos (de MATfor a FORtran).

SUBRUTINAS WHILE: En esta subrutina se efectua la conversión de la proposición "WHILE" encontrada en el programa MATfor a FORtran. Se verifica previamente sintaxis y si dentro de esta proposición no se encuentran llamadas a otras proposiciones o a primitivas.

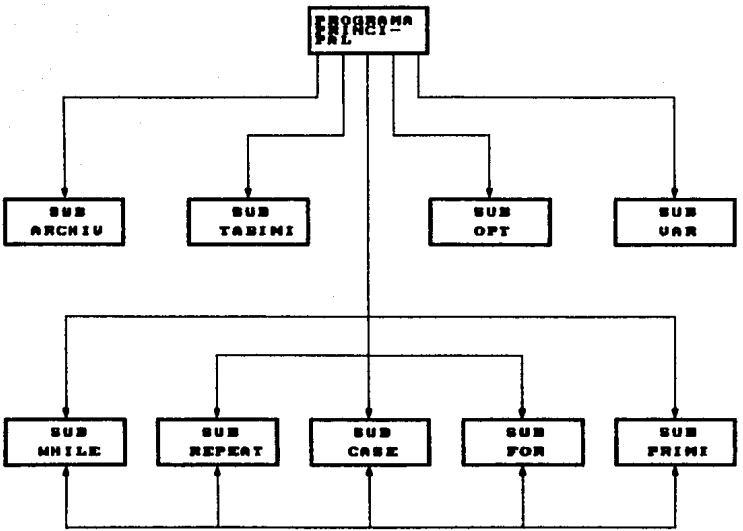
SUBRUTINA REPEAT: Parte donde se realiza la converción de la proposición "REPEAT" a FORtran. También examina si dentro de ella no se hacen llamadas a otras proposiciones o funciones de vectores y matrices.

SUBRUTINA CASE: Subrutina que convierte el "CASE" encontrado en el programa MATfor del archivo de entrada a FORtran, en el archivo de salida. Verifica sintaxis y la existencia de llamadas a otras estructuras de repetición y selección o de alguna primitiva.

SUBRUTINA FOR: Subrutina que convierte la proposición "FOR" a FORtran. Previamente checa sintaxis y posteriormente examina si dentro de ella se llama a otra función disponible en el preprocesador.

SUBRUTINA PRIMI: En esta subrutina se detecta si lo escrito en el programa de entrada, es una primitiva de MATfor. Reconoce la función, verifica sintaxis y realiza su conversión.

PREPROCESADOR MATFOR



Ilustracion 1. Estructura general del Preprocesador MATFOR.

IV.2 DESCRIPCIONES ESPECIFICAS Y DIAGRAMAS

LA SUBRUTINA ARCHIV ESTA LIGADA CON LAS SIGUIENTES SUBRUTINAS

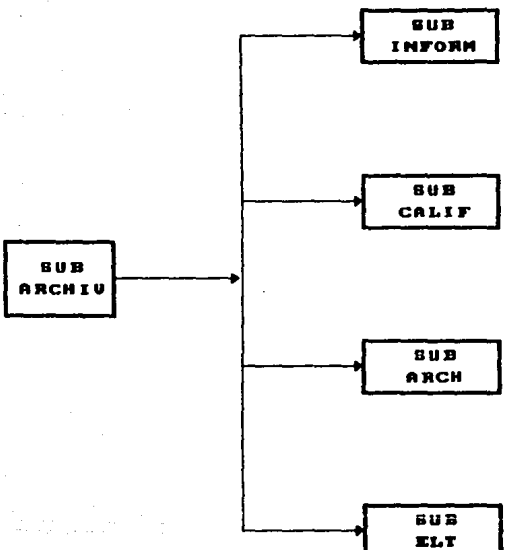
SUBRUTINA INFORM: Rutina creada con el fin de obtener la información proporcionada por el programador en el momento de ejecutar al preprocesador.

SUBRUTINA ARCH: Rutina en ensamblador que se auxilia en la subrutina inform para detectar el primer parámetro mandado en el preprocesador (archivo de entrada).

SUBRUTINA ELT: Esta rutina también se auxilia de inform, está diseñada para obtener el segundo parámetro (archivo de salida).

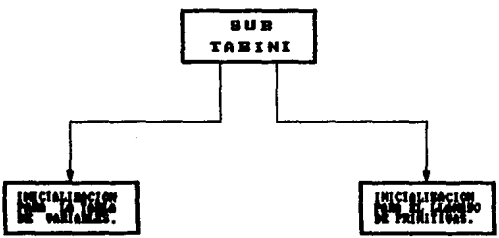
SUBRUTINA CALIF: Rutina creada con el objeto de obtener la cuenta o calificador del usuario que ejecuta el preprocesador, por si lo omite en los archivos pasados como parámetros toma éste por default. Si los archivos no son suyos es necesario que dé el calificador y el nombre del archivo (p.e. calificador$archivo).

CONFIGURACION DEL BLOQUE ARCHIV



Ilustracion 3. Estructura especifica del bloque ARCHIV.

CONFIGURACION DEL BLOQUE TABINI



Ilustracion 3. Estructura especifica del bloque TABINI.

LA SUBRUTINA VAR ESTA LIGADA CON LAS SIGUIENTES SUBRUTINAS

SUBRUTINA MAT: Subrutina utilizada cuando se reconoce una declaración de matriz, en ella se extraen los valores para su dimensionamiento, checa todos los posibles errores y llama a las subrutinas SEPARA y DIMM.

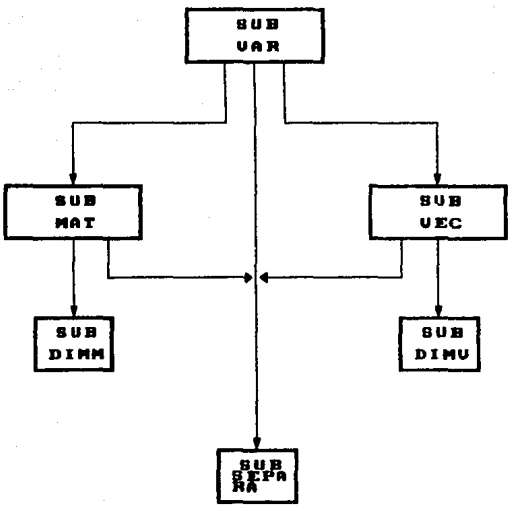
SUBRUTINA VEC: Subrutina a la cual se hace referencia cuando se detecta una declaración para vector, en ella se verifica la sintaxis, busca el número para su dimensionamiento y hace referencia a las subrutinas SEPARA y DIMV.

SUBRUTINA SEPARA: Esta rutina separa las variables que le fueron entregadas por las subrutinas MAT y VEC en una cadena, las mete en un vector y éste es pasado como parámetro a las subrutinas DIMM o DIMV, según sea el caso.

SUBRUTINA DIMM: Subrutina que se encarga de dimensionar con la sintaxis correspondiente, las variables pasadas por la subrutina SEPARA correspondientes a matrices.

SUBRUTINA DIMV: Creada exactamente con los mismos fines de la subrutina DIMM, la diferencia radica en que ésta es para dimensionar vectores.

CONFIGURACION DEL BLOQUE VAR



Ilustracion 4. Estructura especifica del bloque VAR.

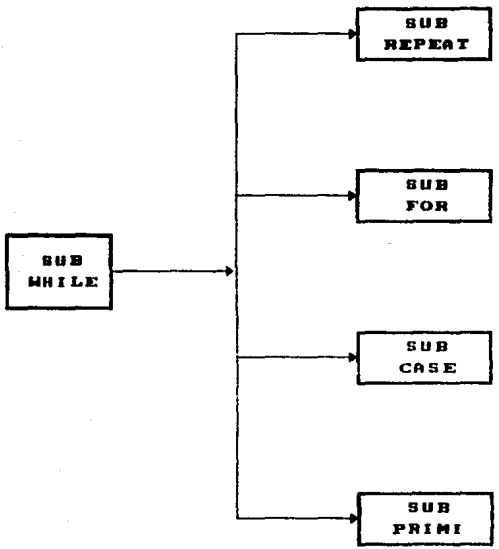
SUBRUTINAS WHILE-REPEAT-CASE-FOR-PRIMI

El objetivo de cada una de estas subrutinas se explicó anteriormente.

La diferencia entre los cinco diagramas siguientes radica sólo en la función de la subrutina principal.

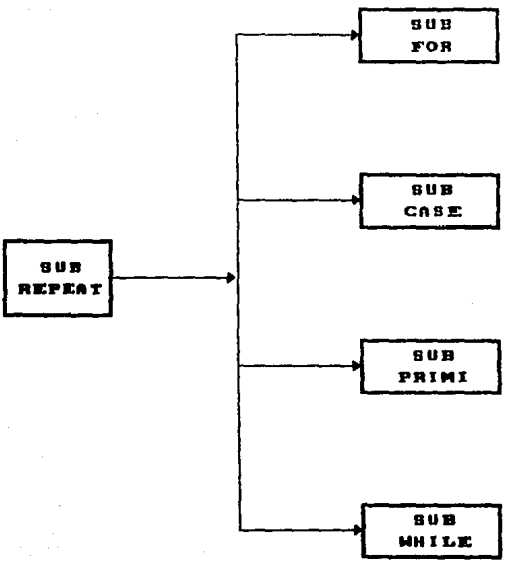
Son cinco subrutinas con fines especificos y con llamadas entre si mismas. Cada una de ellas, cuando le toca estar en la posición principal, checa su sintaxis, lleva a cabo la conversión correspondiente y a la vez verifica si no se le hace referencia a alguna de las cuatro subrutinas restantes.

CONFIGURACION DEL BLOQUE WHILE



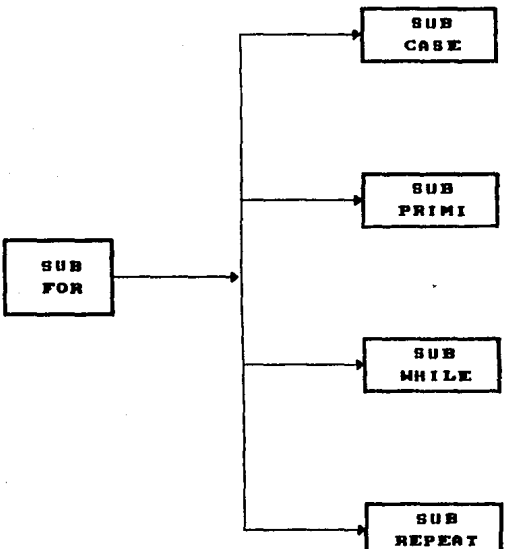
Ilustracion 5. Estructura especifica del bloque WHILE.

CONFIGURACION DEL BLOQUE REPEAT



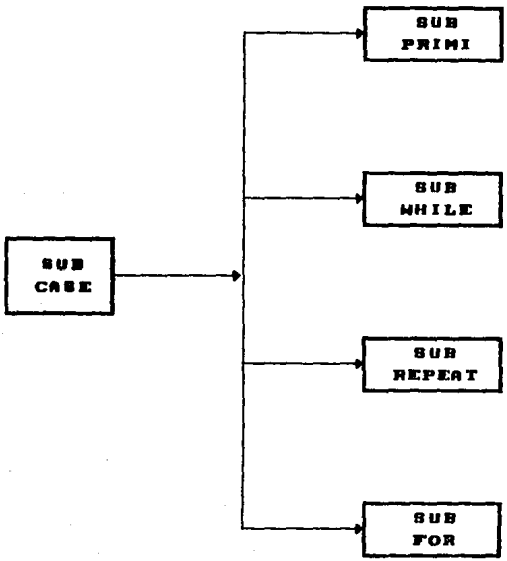
Ilustracion 6. Estructura especifica del bloque REPEAT.

CONFIGURACION DEL BLOQUE FOR



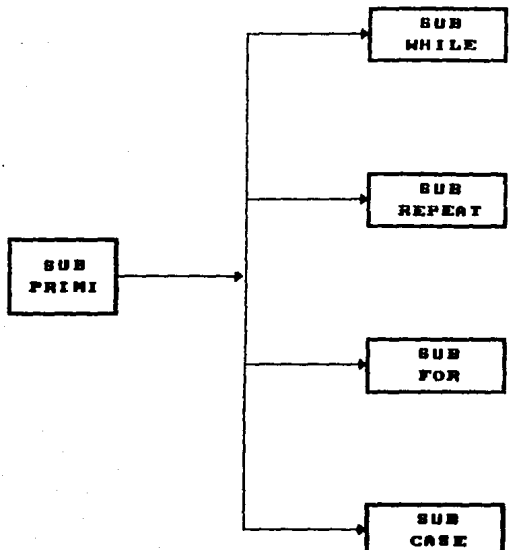
Ilustracion 7. Estructura especifica del bloque FOR.

CONFIGURACION DEL BLOQUE CASE



Ilustracion 8. Estructura especifica del bloque CASE.

CONFIGURACION DEL BLOQUE PRIMI



Ilustracion 9. Estructura especifica del bloque PRIMI

CAPITULO V

"D E S A R R O L L O"

V.1 VARIABLES Y ARREGLOS

En esta sección se lleva a cabo la declaración de variables, así como la de los arreglos a utilizar en un programa que será desarrollado con fines específicos. Se propone una nueva declaración de variables a través de un bloque que nos permita visualizar y comprender mejor esta parte tan importante e inicial de un programa. Cabe señalar que la declaración de variables propuesta es para darle mayor comprensión, flexibilidad y estructuración a dicha declaración, por lo tanto en ningún momento queda excluida la declaración de variables tradicional de FORtran. Es de suma importancia, si se va hacer uso de alguna de las primitivas de MATfor, declarar los escalares, vectores y matrices a utilizar, dentro de la estructura de la declaración de variables propuesta.

V.1.1 FORMATO, ALTERNATIVAS, RESTRICCIONES Y MENSAJES DE ERROR

FORMA GENERAL DEL BLOQUE DE LA DECLARACION DE VARIABLES

VAR

```
Vi[ ,V2, ... ,VN ] : [ NA [ 1..n [ ,1..m ] ] ] T
:
:
Pi[ ,P2, ... ,PN ] : [ NA [ 1..n [ ,1..m ] ] ] T
```

END VAR

En donde

Vi,V2,Pi,P2,... : Nombres de variables mayores o iguales a un dígito y menores que 7.

NA : Nombre del arreglo que puede ser VEC o MAT para vectores y matrices respectivamente.

n,m : Números enteros que especifican el dimensionamiento de las variables.

T : Campo que corresponde al tipo de variable que se desea.

Lo encerrado en los corchetes grandes va para el caso de vectores y matrices.

Lo encerrado en corchetes del lado izquierdo es opcional así como lo que se encuentra en los corchetes más internos del lado derecho.

ALTERNATIVAS

* El bloque para la declaración de variables puede ser abierto y cerrado de las siguientes maneras:

```
VAR                 VAR                 VAR
:                   :                   :
:                   :                   :
END VAR             END V               ENDV

ARRAY               ARRAY               ARRAY
:                   :                   :
:                   :                   :
END ARRAY           END A               ENDA
```

* Se pueden utilizar paréntesis en lugar de los corchetes para dimensionar.

```
VIC ,V2, ... ,VN ] : [ NA(1..n [ ,1..m ] ) ] T
```

* Los tipos de las variables pueden ser escritos de las siguientes posibles maneras:

INTEGER --------- ENTERA - ENTERO - ENTERAS--ENTEROS

REAL ------------ REALES

CHARACTER ------- CARACTER-CHAR

LOGICAL --------- LOGICA - BOOLEAN--LOGICAS

COMPLEX --------- COMPLEJA-COMPLEJAS

DOUBLE PRECISION- DOUBLE

RESTRICCIONES

* Para declarar una variable caracter, dimensionada o no, es necesario que se ponga la variable con el número de digitos a reservar. por ejemplo:

```
VAR

  NOM$30,DIR$12,TEL$7 : CARACTER

  CLA$5 : VEC[1..10] CHARACTER

END VAR
```

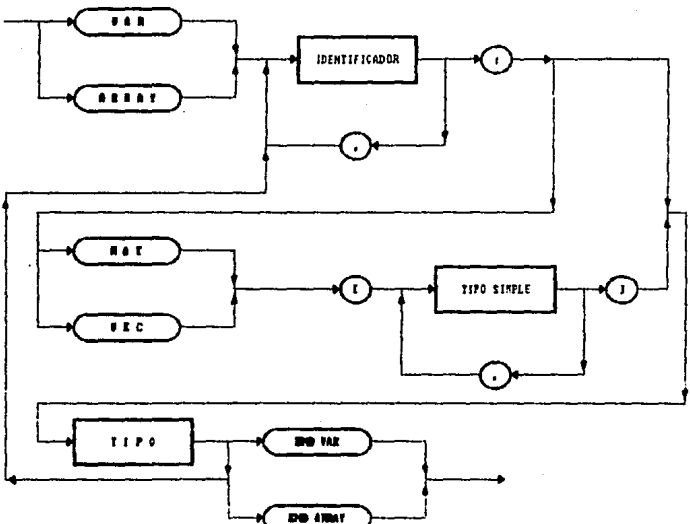
* Si en un renglón, la última variable especificada es de 6 digitos, deberán seguir inmediatamente los dos puntos (es decir, no dejar espacios), porque de lo contrario marcará un error diciendo que la variable es mayor de 6 digitos.

* La longitud de la declaración de una variable tipo caracter (nombre, asterisco y número de posiciones para cada caracter) no debe exceder a 6.

MENSAJES DE ERROR

```
ERROR!!... FALTARON (:) EN DEC. "VAR"

ERROR!!... "VAR" O "ARRAY" SIN TIPO
           CHECAR SINTAXIS


ERROR!!... EN DECLARACION DE MATRICES
           MAT[1<.>N,1..M] ---> MAT[1..N,1..M]

ERROR!!... EN DECLARACION DE MATRICES PUNTO DE MAS
           MAT[1<...>N,1..M] ---> MAT[1..N,1..M]

ERROR!!... EN DECLARACION DE MATRICES
           MAT[1..N,1<.>M] ---> MAT[1..N,1..M]

ERROR!!... EN DECLARACION DE MATRICES PUNTO DE MAS
           MAT[1..N,1<...>M] ---> MAT[1..N,1..M]

ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "MAT"

ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "MAT"

ERROR!!... EN "MAT" TIPO DE VARIABLE NO CONSIDERADO


ERROR!!... EN DECLARACION DE VECTORES
           VEC[1<.>N] ---> VEC[1..N]

ERROR!!... EN DECLARACION DE VECTORES PUNTO DE MAS
           VEC[1<...>N] ---> VEC[1..N]

ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "VEC"

ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "VEC"

ERROR!!... EN "VEC" TIPO DE VARIABLE NO CONSIDERADO


ERROR!!... SINTAXIS EN DEC. DE "VAR" O "ARRAY"

ERROR!!... EN "VAR" O "ARRAY" VARIABLE MAYOR DE 6

ERROR!!... EN DECLARACION DE "VAR" O "ARRAY"
```

REPRESENTACION GRAFICA
DECLARACION DE VARIABLES



V.2 PROPOSICION WHILE

La proposición WHILE es una proposición parecida a la de PASCal, nos sirve para tener en forma estructurada un ciclo de repetición, la condición se evalua al principio y si es falsa se sale del ciclo, en caso contrario se queda ejecutando una o un conjunto de instrucciones que componen el bloque para dicho ciclo.

En la programación práctica , la proposición WHILE es mucho más útil que la proposición REPEAT. Esto se debe al hecho de que en la mayoria de los casos la posibilidad de que el ciclo pueda no ser ejecutado, debe ser reconocida y tomada en cuenta

Es necesario que la condición tenga un valor bien definido al entrar a la declaración. Es esencial que la instrucción dentro del ciclo repetitivo eventualmente cambie el valor de la condición porque de otra manera el ciclo repetitivo se continurá ejecutando idefinidamente.

V.2.1 FORMATO, ALTERNATIVAS, RESTRICCIONES Y MENSAJES DE ERROR

FORMA GENERAL DE LA PROPOSICION WHILE

El formato general para esta proposición es el siguiente:

WHILE (condición) DO

DECLARACION

Donde la DECLARACION puede ser una instrucción o un bloque de instrucciones. Para el segundo caso es necesario abrir el bloque con la palabra **BEGIN** y cerrarlo con las palabras **END WHILE.**

ALTERNATIVAS

* La terminación del bloque abierto en el WHILE puede ser de las siguientes maneras:

```
WHILE                WHILE                WHILE
  BEGIN                BEGIN                BEGIN
    :                    :                    :
    :                    :                    :
END WHILE            END W                ENDW
```

* Entre el bloque del WHILE pueden existir líneas de comentarios (cualquier caracter que no sea número entre las columnas de la 1 a la 5), así como líneas de continuación (cualquier caracter en la columna 6). También es posible poner el caracter de reconocimiento de función acompañado de una etiqueta (!etiq).

RESTRICCIONES

* Las variables a utilizar en la condición deben de ser de un máximo de 6 dígitos.

* La condición debe de ir entre paréntesis solamente.

* Los operadores relacionales y lógicos deben de ser los mismo de FORtran y son los siguientes:

OPERADOR	USO	EXPLICACION
.GT.	e1.GT.e2	Verdad si e1 es mayor que e2.
.GE.	e1.GE.e2	Verdad si e1 es mayor o igual que e2.
.LT.	e1.LT.e2	Verdad si e1 es menor que e2.
.LE.	e1.LE.e2	Verdad si e1 es menor o igual que e2.
.EQ.	e1.EQ.e2	Verdad si e1 es igual que e2.
.NE.	e1.NE.e2	Verdad si e1 no es igual a e2
.NOT.	.NOT.e1	La expresión tiene la oposición lógica de la exp. e1.
.AND.	e1.AND.e2	La expresión es verdadera si ambos e1 y e2 son verdaderos, en otro caso es falsa.
.OR.	e1.OR.e2	Si e1,e2 o ambos son verdaderos la expresión será verdadera, de lo contrario falsa.

MENSAJES DE ERROR

ERROR!!...FALTO ESPACIO ENTRE EL WHILE Y LA CONDICION
("WHILE (CONDI.)")

ERROR!!... FALTO CONDICION EN EL "WHILE"

ERROR!!... MALA DECLARACION DEL "WHILE" FALTO "DO"

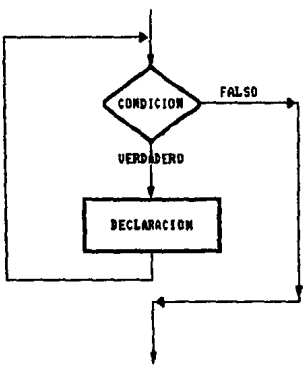
ERROR!!... FALTO ESPACIO ENTRE "CONDICION" Y "DO"

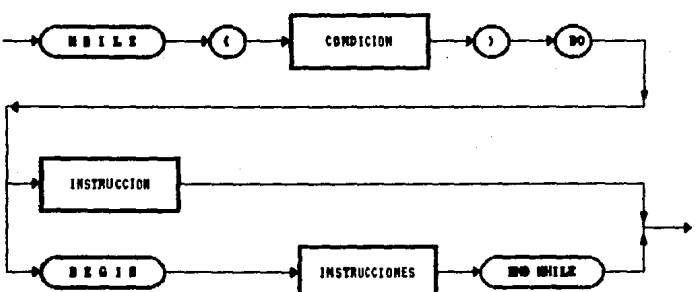
ERROR!!... CONDICION DEL "WHILE" VA ENTRE PARENTESIS

ERROR!!... PARENTESIS INCOMPLETOS EN EL "WHILE"

ERROR!!... FALTO "ENDW" O "END WHILE"

DIAGRAMA DE FLUJO Y REPRESENTACION GRAFICA PROPOSICION WHILE





V.3 PROPOSICION REPEAT

La proposición REPEAT tiene dos partes: el ciclo y la condición de terminación.

La proposición REPEAT es usada cuando no se sabe a la hora de escribir un programa, cuantas repeticiones serán necesarias.

Existen tres casos a considerar al escribir el ciclo repetitivo REPEAT:

a) Las condiciones iniciales deben de ser correctas

b) Las instrucciones dentro del ciclo deben estar correctamente secuenciadas y debe haber al menos una que tenga efecto sobre la condición de terminación.

c) La condición de terminación debe eventualmente ser satisfecha. De otra manera el ciclo continuará indefinidamente.

Esta proposición de repetición presentada es parecida a la de PAScal. Primero se presenta la declaración (una o varias instrucciones) a ejecutarse dentro del ciclo y, por último, se evalúa la condición de terminación, si ésta es falsa se ejecuta nuevamente el ciclo y si es verdadera sale de él.

V.3.1 FORMATO, ALTERNATIVAS, RESTRICCIONES Y MENSAJES DE ERROR

FORMATO GENERAL DE LA PROPOSICION REPEAT

La sintaxis general de la proposición REPEAT se presenta a continuación :

```
REPEAT
  DECLARACION
UNTIL (condición)
```

En la DECLARACION puede ir una instrucción o un conjunto de instrucciones.

ALTERNATIVAS

§ La proposición REPEAT tiene las siguientes posibilidades de declaración :

```
REPEAT              REPEAT              REPEAT
  :                   :                   :
  :                   :                   :
UNTIL (condi.)      UNT (condi.)        UNTIL(condi.)
```

§ Entre el bloque del REPEAT pueden existir lineas de comentarios (cualquier caracter que no sea número entre las columnas de la 1 a la 5), asi como lineas de continuación (cualquier caracter en la columna 6). También es posible poner el caracter de reconocimiento de función acompañado de una etiqueta (!etiq).

RESTRICCIONES

§ Las variables a utilizar en la condición deben de ser menores o iguales a 6 dígitos.

§ La condición debe de ir únicamente entre paréntesis.

§ Los operadores relacionales y lógicos deben de ser los mismo de FORTran y son los siguientes:

OPERADOR	USO	EXPLICACION
.GT.	e1.GT.e2	Verdad si e1 es mayor que e2.
.GE.	e1.GE.e2	Verdad si e1 es mayor o igual que e2.
.LT.	e1.LT.e2	Verdad si e1 es menor que e2.
.LE.	e1.LE.e2	Verdad si e1 es menor o igual que e2.
.EQ.	e1.EQ.e2	Verdad si e1 es igual que e2.
.NE.	e1.NE.e2	Verdad si e1 no es igual a e2
.NOT.	.NOT.e1	La expresión tiene la oposición lógica de la exp. e1.
.AND.	e1.AND.e2	La expresión es verdadera si ambos e1 y e2 son verdaderos, en otro caso es falsa.
.OR.	e1.OR.e2	Si e1,e2 o ambos son verdaderos la expresión será verdadera, de lo contrario falsa.

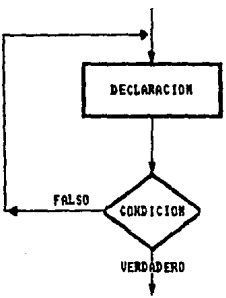
MENSAJES DE ERROR

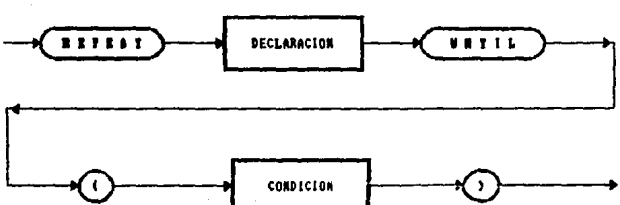
ERROR!!... PARENTESIS INCOMPLETOS EN EL "REPEAT-UNTIL"

ERROR!!... FALTO CONDICION DEL "REPEAT-UNTIL"

ERROR!!... NO SE ENCONTRO "UNTIL" DEL "REPEAT"

DIAGRAMA DE FLUJO Y REPRESENTACION GRAFICA PROPOSICION REPEAT





V.4 PROPOSICION FOR

Cuando deseamos ejecutar una proposición repetitivamente, y el número de repeticiones no depende del efecto de las instrucciones dentro del ciclo, entonces la construcción apropiada es el ciclo FOR.

Este tiene los mismos principios que el DO de FORtran. El FOR se presenta aquí para ampliarle al programador las instrucciones para realizar acciones de repetición, dejandole a su criterio el uso de una o de la otra.

El hecho de que una proposición FOR pueda ser escrita en la forma de una proposición WHILE, implica que la proposición FOR es redundante. Sin embargo, hay buenas razones para usar la proposición FOR en donde sea posible. La proposición FOR proporciona más información al lector. Los valores que han de asignarse a la variable de control, y el número de veces que será ejecutado el ciclo son, ambos, captados de inmediato. La misma información es también útil al compilador, el cual, con frecuencia, podrá producir un programa más eficiente a partir de una proposición FOR, que a partir de la proposición WHILE equivalente.

V.4.1 FORMATO, ALTERNATIVAS, RESTRICCIONES Y MENSAJES DE ERROR

FORMATO GENERAL DE LA PROPOSICION FOR

La sintaxis general de la proposición FOR es:

FOR vc = exp1 **TO** exp2 **DO**

DECLARACION

En la DECLARACION puede ir una instrucción o un conjunto de instrucciones. En el segundo caso se requiere abrir y cerrar el bloque que contendrá ese conjunto de instrucciones, con BEGIN y END FOR respectivamente.

La proposición sólo tendrá efecto si :

exp1 < exp2

La palabra clave TO en la proposición FOR puede ser reemplazada por DOWNTO. La proposición FOR se convierte en:

FOR vc = exp1 **DOWNTO** exp2 **DO**

DECLARACION

En éste caso, en lugar de aumentar, la variable de control se decrementa en cada iteración. Esta forma de la proposición FOR tendrá efecto si :

exp1 > exp2

ALTERNATIVAS

La proposición FOR tiene las siguientes alternativas:

* El valor inicial de la variable de control puede ser asignado de las siguientes maneras:

FOR vc = exp1 TO exp2 DO

FOR vc := exp1 TO exp2 DO

* El bloque abierto para el FOR puede ser terminado de las siguientes maneras:

```
FOR                FOR                FOR
  BEGIN              BEGIN              BEGIN
    :                  :                  :
    :                  :                  :
END FOR            END F              ENDF
```

* Entre el bloque del FOR pueden existir líneas de comentarios (cualquier caracter que no sea número entre las columnas de la 1 a la 5), así como líneas de continuación (cualquier caracter en la columna 6). También es posible poner el caracter de reconocimiento de función acompañado de una etiqueta, (!etiq).

RESTRICCIONES

* Si las expresiones exp1 y exp2 no son números enteros y son variables, se tiene que cuidar que no sean mayores de 6 dígitos.

* Si se desea que más de una instrucción se realice en la proposición FOR, es necesario abrir un bloque con BEGIN y cerrarlo con END FOR, porque de lo contrario sólo ejecutará la primera instrucción

* La variable de control del FOR tiene que ser mayor o igual que 1 y menor de 7 dígitos.

MENSAJES DE ERROR

ERROR!!... MALA DECLARACION PARA EL "FOR"

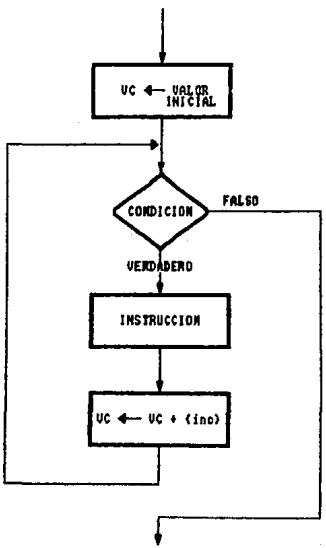
ERROR!!... VARIABLE MAYOR DE 6 DIGITOS EN EL "FOR"

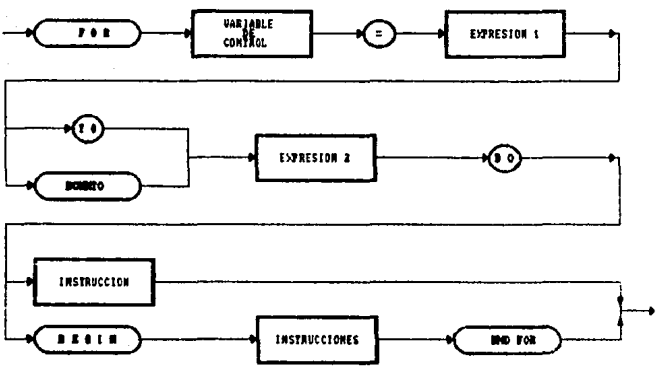
ERROR!!... DECLARACION DEL "FOR" INCOMPLETA FALTO "DO"

ERROR!!... FALTO ESPACIO PARA EL "DO" EN EL "FOR"

ERROR!!... NO SE ENCONTRO "ENDF" O "END FOR"

DIAGRAMA DE FLUJO Y REPRESENTACION GRAFICA PROPOSICION FOR




FOR
VARIABLE DE CONTROL
=
EXPRESION 1
TO
DOWNTO
EXPRESION 2
DO
INSTRUCCION
BEGIN
INSTRUCCIONES
END FOR

V.5 PROPOSICION CASE

La proposición IF permite a un proceso optar por una de las dos posibles alternativas de acción, de acuerdo al valor de una expresión booleana.

El CASE es una generalización de la declaración IF: ésta permite al proceso ejecutar una de varias acciones de acuerdo con el valor de una expresión escalar o subintervalo. El CASE aquí presentado nos permitirá escribir un programa en una forma más ordenada a la que resultaría si utilizaramos puros IF, volverá más flexible la programación y será más entendible para cualquier lector.

Se muestra un ejemplo para entender mejor como esta estructurado el CASE: Supongamos que una compañía tiene 3 máquinas y se les renta al público con tarifas diferentes:

```
CASE equip OF

  pc :
       tarifa = 1000

  mini :
       tarifa = 10000

  mframe :
       tarifa = 100000

END CASE
```

Cuando se requiere la misma acción para valores diferentes de la variable en cuestión, estos valores pueden escribirse en una lista: por ejemplo, los días que corresponden a cada mes :

```
CASE mes OF

  ene,mar,may,jul,ago,oct,dic :
                                    dias = 31

  abr,jun,sep,nov :
                                    dias = 30

END CASE
```

NOTA: Febrero no se metió, se necesita un algoritmo para saber si es de 28 o de 29 días y no es el objetivo de esta sección.

Cuando se necesitan más de una acción para un valor o diferentes valores de la variable en cuestión, se logra abriendo y cerrando un subbloque. Por ejemplo:

```
CASE vc OF

  1,7,11 : instrucción1

       2 : BEGIN
             instrucción2
             instrucción3
             instrucción4
           END BEGIN

  4,6,8, : BEGIN
             instrucción5
             instrucción6
           END BEGIN

END CASE
```

Las palabras reservadas CASE y END CASE actuan como paréntesis alrededor de la proposición.

V.5.1 FORMATO, ALTERNATIVAS, RESTRICCIONES Y MENSAJES DE ERROR

FORMA GENERAL DE LA PROPOSICION CASE

El formato general para esta proposición es el siguiente:

```
CASE vc OF

  expresión : declaración

END CASE
```

Donde

expresión : puede ser un expresión o varias separadas por comas.

declaración: pude ser una instrucción o varias encerradas en el subbloque BEGIN END BEGIN.

ALTERNATIVAS

* La proposición CASE puede ser terminada de las siguientes maneras:

```
CASE                CASE                CASE
  :                   :                   :
  :                   :                   :
END CASE            END C               ENDC
```

* Si se abre un subbloque dentro del CASE, éste puede ser terminado de tres maneras:

```
CASE                CASE                CASE
  :                   :                   :
BEGIN               BEGIN               BEGIN
  :                   :                   :
  :                   :                   :
END BEGIN           END B               ENDB
  :                   :                   :
END CASE            END CASE            END CASE
```

* Entre el bloque del CASE pueden existir líneas de comentarios (cualquier caracter que no sea número entre las columnas de la 1 a la 5), así como líneas de continuación (cualquier caracter en la columna 6). También es posible poner el caracter de reconocimiento de función acompañado de una etiqueta (!etiq).

* Se puede poner el elemento de un vector o de una matriz como variable en cuestión. Inclusive una expresión compleja, cuidando de cerrar bien los paréntesis abiertos.

RESTRICCIONES

* Si la variable en cuestión es de tipo caracter, las expresionesde comparación deben de estar entre apóstrofes.
Por ejemplo:

```
'MES','DIA','ANO' :
```

* No se permiten espacios en blanco entre los apóstrofes.
Por ejemplo:

```
'MESES','DIAS','ANOS'     correcta

'MESES','DIAS ','ANOS '   incorrecta
```

* La variable en cuestión tiene que ser igual o menor a 6 dígitos.

MENSAJES DE ERROR

ERROR!!... MALA DECLARACION DEL "CASE" FALTO ESPACIO

ERROR!!... MALA DECLARACION DEL "CASE", $VARIABLE EN CUESTION??$

ERROR!!... VARIABLE EN CUESTION DEL "CASE" MAYOR DE 6 DIGITOS

ERROR!!... FALTO ESPACIO PARA EL "OF" EN EL "CASE"

ERROR!!... PARENTESIS INCOMPLETOS EN EL "CASE"

ERROR!!... MALA DECLARACION DEL "CASE" FALTO "OF"

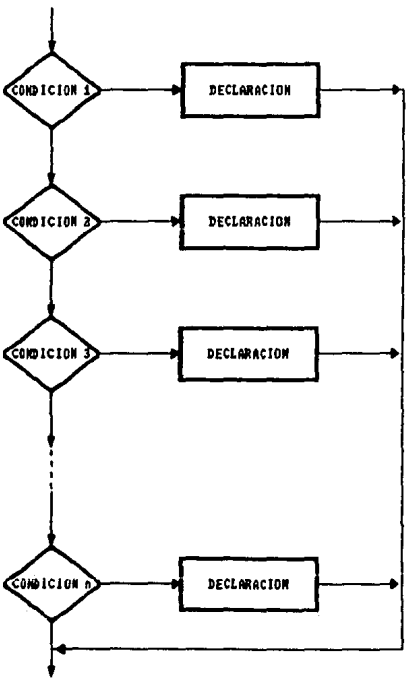
ERROR!!... FALTO DECLARACION EJECUTABLE DESPUES DE LOS DOS PUNTOS

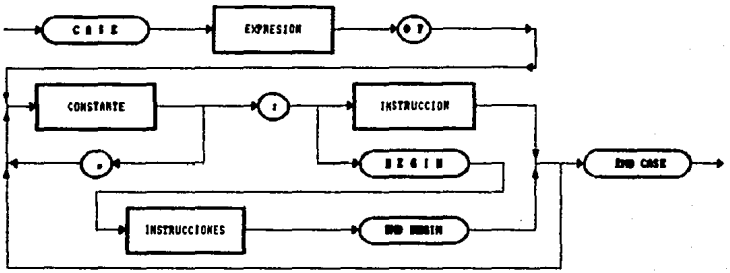
ERROR!!... FALTARON LOS DOS PUNTOS (:) O "END CASE"

ERROR!!... FALTO "END CASE"

ERROR!!... FALTO CERRAR EL BEGIN ABIERTO "END BEGIN"

DIAGRAMA DE FLUJO Y REPRESENTACION GRAFICA
PROPOSICION CASE




CASE
EXPRESION
OF
CONSTANTE
:
INSTRUCCION
,
BEGIN
INSTRUCCIONES
END MAIN
END CASE

V.6 VECTORES Y MATRICES

Un **vector** es un conjunto de valores asociados a un nombre común y diferenciados por un índice.

Una **matriz** es una disposición rectangular de números que tiene "n" filas y "m" columnas.

Un **escalar** es un número real o entero.

Las operaciones entre vectores, matrices y escalares, para algunos resulta un poco complicado por el manejo de los subíndices, y más aun cuando se requieren implementar estos algoritmos en computación.

Existen dos tipos de operadores para el uso de las funciones implementadas a la manipulación de vectores, matrices y escalares, y són ¦ Los operadores Monádicos y los operadores Diádicos.

Si un operador actúa sobre dos variables, efectúa una operación diádica. Suponga que la operación es "A-B", entonces el operador (-) se usa como operador diádico. Por otra parte si se escribe ">A" el signo (>) se está utilizando como un operador monádico.

V.6.1 FUNCIONES PARA VECTORES

Con las funciones implementadas para el uso de vectores, el programador se ahorrará escribir más líneas de código, le simplificará información redundante y le evitará tener en forma explícita varios ciclos.

Existen trece funciones para las operaciones entre vectores y son:

1) Lectura de un vector (horizontalmente).
2) Lectura de un vector (verticalmente).
3) Impresión de un vector (h.).
4) Impresión de un vector (v.).
5) Suma de dos vectores.
6) Resta de dos vectores.
7) Multiplicación de dos vectores.
8) Multiplicación de un vector por una matriz.
9) Multiplicación de un vector por un escalar.
10) Multiplicación de un escalar por un vector.
11) Máximo de un vector.
12) Mínimo de un vector.
13) Medio de un vector.

V.6.2 FORMATOS PARA VECTORES

Acontinuación se presentan las sintaxis para las funciones de vectores.

FUNCION	SINTAXIS	OPERADOR
Lectura de un vector (h)	?V	Monádico
Lectura de un vector (v)	?V@	Monádico
Impresión de un vector (h)	&V	Monádico
Impresión de un vector (v)	&V@	Monádico
Suma de dos vectores	V=V1+V2	Diádico
Resta de dos vectores	V=V1+v2	Diádico
Mul. de dos vectores	V=V1&V2	Diádico
Mul. de un VEC por una MAT	V=V&M	Diádico
Mul. de un VEC por un ESC	V=V&E	Diádico
Mul. de un ESC por un VEC	V=E&V	Diádico
Máximo de un vector	E=>V	Monádico
Mínimo de un vector	E=<V	Monádico
Medio de un vector	E=@V	Monádico

Donde

V,V1,V2 : Son nombres de variables vectoriales.

E : Nombre de una variable escalar.

M : Nombre de una matriz.

V.6.3 FUNCIONES PARA MATRICES

La manipulación de matrices en el escritorio, a veces resulta tediosa y es necesario hacer algoritmos computacionales para simplificar este uso y tener más confiabilidad en los resultados. En esta sección se simplifica aún más estos requerimientos, ya que con el simple hecho de invocar a las funciones tenemos lo deseado y el programador se olvida de desarrollar los mencionados algorítmos. Esto resulta más útil, más productivo, mejor comprensión de los programas para cualquier lector y le reduce un sinnúmero de líneas de código.

Se han desarrollado 9 funciones para la operación de matrices. A continuación se mencionan dichas funciones:

1) Lectura de una matriz.

2) Impresión de una matriz.

3) Suma de dos matrices.

4) Resta de dos matrices.

5) Multiplicación de dos matrices.

6) Multiplicación de una matriz por un vector.

7) Multiplicación de una matriz por un escalar.

8) Multiplicación de un escalar por una matriz.

9) Inversa de una matriz.

V.6.4 FORMATOS PARA MATRICES

La sintaxis para el conjunto de funciones de matrices es la siguiente:

FUNCION	SINTAXIS	OPERADOR
Lectura de una matriz	?M	Monádico
Impresión de una matriz	&M	Monádico
Suma de dos matrices	M=M1+M2	Diádico
Resta de dos matrices	M=M1-M2	Diádico
Mul. de dos matrices	M=M1&M2	Diádico
Mul. de una MAT por un VEC	V=M&V1	Diádico
Mul. de una MAT por un ESC	M=M*E	Diádico
Mul. de un ESC por una MAT	M=E&M	Diádico
Inversa de una matriz	M=@M	Monádico

Donde

M,M1,M2 : Son nombres de variables que corresponden a matrices.

E : Nombre de una variable escalar.

V : Nombre de una vector.

V.6.5 RESTRICCIONES Y MENSAJES DE ERROR

* Al hacer uso de una de estas funciones, hay que tener previamente definidas las variables a utilizar en el bloque designado para la declaración de variables.

* Aunque se hace previamente el chequeo de la longitud de las variables, no está por demás recordar que no tienen que exceder a 6 dígitos.

* Las matrices utilizadas para inversión deben de ser del tipo real.

* También es necesario recordar que la forma de indicarle al preprocesador que se trata de una función propia, es a través del signo de admiración (!), que es el caracter de reconocimiento de función y va colocado entre las columnas de la uno a la cinco.

* Si uno se llega a equivocar al invocar una función propia de MATfor, no hay ningun problema, ya que el preprocesador se lo hará saber inmediatamente.

Los mensajes de error que se manejan en esta sección son los siguientes:

```
ERROR!!... FUNCION NO DISPONIBLE

ERROR!!... DIMENSIONES INCOMPATIBLES EN
           "V=V+V,V=V-V,V=V$V"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V$M"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V$E"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=E$V"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M+M O M=M-M"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M$M"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=M$V"

ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M$E"
```

```
ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=E*M"

ERROR!!... DIMENSIONES INCOMPATIBLES
           MATRIZ CUADRADA PARA OBTENER SU INVERZA

ERROR!!... FUNCION NO CONSIDERADA EN SUB-COMPAT

ERROR!!... FUNCION NO CONSIDERADA

ERROR!!... VARIABLE MAYOR DE 6 DIGITOS
           AL USAR FUNCION EXCLUSIVA DE *MATFOR*

ERROR!!... FUNCION DE *MATFOR* DECLARADA INCOMPLETA

ERROR!!... FUNCION UNIVOCA DE *MATFOR* MAYOR DE 6
           DIGITOS

ERROR!!... NO SE ENCONTRO SIGNO DE (=) EN FUNCION DE
           *MATFOR* O > BLANCOS

ERROR!!... FUNCION DE *MATFOR* DECLARADA INCOMPLETA
           CAMPO DE VARIABLE EN BLANCO

ERROR!!... VARIABLE NO DECLARADA EN "VAR" O "ARRAY"
           Y UTILIZADA EN FUNCIONES DE *MATFOR*

ERROR!!... VARIABLES DE DIFERENTE TIPO EN FUNCIONES DE
           *MATFOR* REALES/ENTERAS???
```

V.7 PROPOSITO Y MANIPULACION DE TABLAS

El uso de tablas en el desarrolo de este preprocesador, tiene el objetivo de darle mejor velocidad a la traslación de información del archivo de entrada, al de salida.

Tenemos en total 5 tablas :

a) Tabla de variables.

b) Tabla de funciones.

c) Tabla de compatibilidad.

d) Tabla de estatus.

e) Tabla de bibliotecas.

a) La tabla de variables, tiene el propósito de guardar las variables declaradas al inicio del programa con sus respentivas dimensiones y el tipo correspondiente a cada variable.

La tabla de variables está compuesta por una matriz de 100 filas y 5 columnas. Sus campos tienen los siguientes nombres de referencia.

NOMBRE DE LA VARIABLE	ARREGLO O ESCALAR	DIMEN SION1	DIMEN SION2	TIPO
. . .	. . .	. . .	. . .	. . .

Por ejemplo:

DADO	V	5		R
TAB	M	2	3	E
. . .	. . .	. . .	. . .	. . .
UNO	E			R

El primer registro se refiere a la variable DADO, que es un vector de 5 campos y del tipo real

El segundo almacena información correspondiente a la variable TAB, que es una matriz de 2 x 3 de tipo entero.

El último registro corresponde a un escalar de nombre UNO, que es del tipo real.

b) En la tabla de funciones, se tienen almacenadas las 22 funciones manejables por el preprocesador. Esta tabla esta compuesta por una matriz de 22 x 2 y los nombres de los campos son los siguientes:

FUNCION FORMADA	NOMBRE DE LA FUNCION
. . .	. . .

Por ejemplo:

V = V + V	SUMVEC
&M	IMPMAT
. . .	. . .
V = M * V	MMVMAT

En el primer registro se grabó la función formada de la instrucción leida por el preprocesador, corresponde a una suma de vectores y es una función disponible en el preprocesador; en el segundo campo del registro se encuentra el nombre de la función.

En el segundo registro se almacenó la función correspondiente a la impresión de una matriz.

El último registro contiene una función formada por el preprocesador, de acuerdo a la instrucción leida del programa de entrada, es una función disponible; y en el segundo campo del registro, se encuentra el nombre de identificación interna de la función. La función es multiplicar una matriz por un vector.

En el primer campo se puede llegar a formar una función no disponible, en este caso el preprocesador se lo hará saber de inmediato al programador.

c) En la tabla de compatibilidad, se verifica valga la redundancia, la compatibilidad de las variables que formaron una función disponible en el preprocesador.

FUNCION FORMADA	ALGORITMO DE COMPATIBILIDAD
. . .	. . .

(Anque no es ésta propiamente una tabla, se manejó así para hacer más entendible esta sección del desarrollo del preprocesador.)

d) La tabla de estatus, tiene como propósito, ir registrando el pedido de las funciones. Cuando se pide por primera vez una función, se prende una bandera y se insertan sus líneas de código, cuando es pedida por segunda vez, se verifica si la bandera está prendida y en este caso ya no inserta las líneas de código. De esta manera se controla la duplicidad de líneas.

La tabla está compuesta de una matriz de 22 x 2 y sus columnas tienen los siguientes nombres:

BANDERA	NOMBRE DE LA FUNCION
. . .	. . .

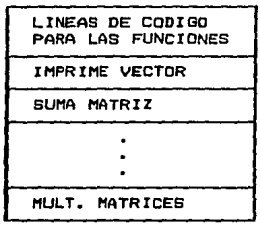
Por ejemplo:

0	INVMAT
. . .	. . .
1	LEEVEC

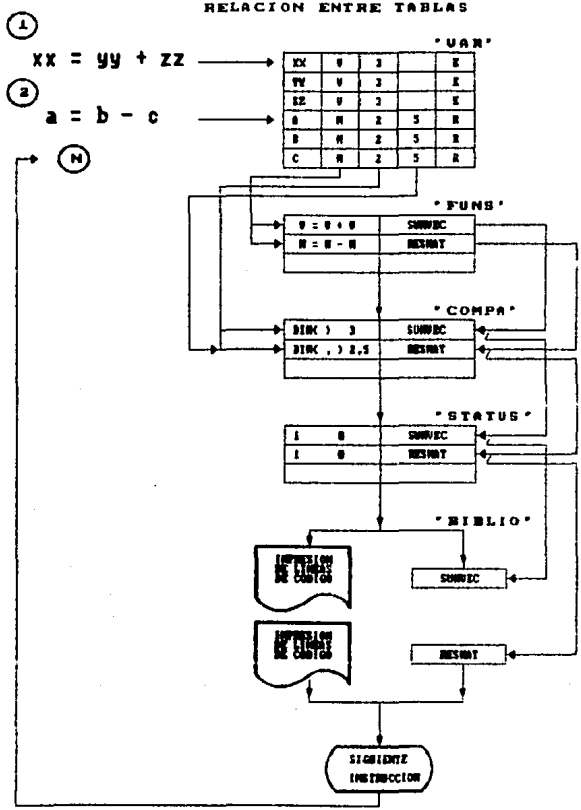
El primer registro que se refiere a la función de invertir una matriz, no ha sido pedida al preprocesador, en el momento que se le pida pondra un 1 en el primer campo e insertará las líneas de código correspondientes a esta función, como una subrutina en el archivo de salida.

El último registro corresponde a la lectura de un vector, el primer campo tiene el número 1, esto indica que ya fue pedida anteriormente esta función, por lo tanto ya existen las líneas de código correspondientes a ella (como subrutina), el preprocesador sólo pondrá el llamado (call) con sus respectivos parámetros.

e) En la tabla de bibliotecas, se encuentran las líneas de código correspondientes a las 22 funciones. Estas líneas son las que se insertan como subrutinas en el archivo de salida, dependiendo del primer campo de la tabla de estatus.



8 La relación entre las tablas es la siguiente:


RELACION ENTRE TABLAS
1
XX = yy + zz
2
a = b - c
N
'VAR'
XX
YY
ZZ
A
B
C
'FUNS'
'COMPA'
'STATUS'
'BIBLIO'
SIGUIENTE INSTRUCCION

V.8 IMPLEMENTACION

Acontinuación se presentan los programas que integran al preprocesador.

$ PARTE 1$ Programa principal y subrutinas mas afines.

KBS$$BCTE$.PREPRO

```
      IMPLICIT INTEGER(A-Z)
      COMMON /VARBLE/VARS
      COMMON /ETIKET/ETI1,ETI2,DOCE
      CHARACTER L1$6,L2$74,VARS$6(100,5),LN$80
      CHARACTER$80 ARCH1,ARCH2,FQA$12,CAR$1
C     ESTE PROGRAMA ES UN PRECOMPILADOR QUE AYUDA AL COMPILADOR
C     DE 'FORTRAN' A ACEPTAR ESTRUCTURAS DE 'PASCAL', EN EL MIS
C     MO PROGRAMA FORTRAN, ASI COMO DIVERSAS FUNCIONES PARA EL
C     USO DE VECTORES Y MATRICES.
      DO 1 I=1,80
      SUBSTR(ARCH1,I,1) = ' '
    1 SUBSTR(ARCH2,I,1) = ' '
      CALL ARCHIV(ARCH1,ARCH2,FQA)
      CALL TABINI
      ETI1=79999
      ETI2=ETI1
      DOCE=0
      CALL OPT(M)
      IF(BITS(M,11,1).NE.1)THEN
        IF(BITS(M,29,1).NE.1)THEN
          STOP ' ERROR!!... OPCION INCORRECTA (MATFOR,ABS O
     $MATFOR,SYM)'
        END IF
      END IF
C     LA SUBRUTINA 'FACSF' SIRVE PARA PODER ASIGNAR TARJETAS DE
C     CONTROL EN UN PROGRAMA.
      IF(ARCH1.EQ.ARCH2)THEN
        ARCH2='
        GO TO 2
      END IF
      IL=INDEX(ARCH2,'.')
      IERR=FACSF2('@ASG,A '//ARCH2(1:IL)//' . ')
      IF(IERR.NE.0.AND.IERR.NE.8589934592)THEN
        WRITE(6,'(2X,'' ERROR!!... ARCHIVO '',A,'' NO CATALOGADO.
     $'')')ARCH2(1:IL)
        STOP
      ELSE
         IERR=FACSF2('@FREE '//ARCH2(1:IL)//' . ')
      END IF
    2 IL=INDEX(ARCH1,'.')
      IERR=FACSF2('@ASG,A '//ARCH1(1:IL)//' . ')
```

```
      IF(IERR.NE.0.AND.IERR.NE.8589934592)THEN
       WRITE(6,'(2X,'' ERROR!!... ARCHIVO '',A,'' NO CATALOGADO.
     *'')')ARCH1(1:IL)
        STOP
      END IF
      IM=INDEX(FQA,' ')
      IERR= FACSF2('@USE 10.,'//ARCH1(1:IL)//' . ')
      CALL FACSF('@ASG,T '//FQA(1:IM-1)//'*MATFOR. . ')
      CALL FACSF('@USE 11.,'//FQA(1:IM-1)//'*MATFOR. . ')
      CALL FACSF('@ASG,T '//FQA(1:IM-1)//'*BIBLIO. . ')
      CALL FACSF('@USE 12.,'//FQA(1:IM-1)//'*BIBLIO. . ')
      IF(BITS(M,11,1).EQ.1)THEN
        WRITE(11,'(''@ASG,T FOREXT.'')')
        WRITE(11,'(''@BRKPT PRINT$/FOREXT'')')
        WRITE(11,'(''@FTN,S'')')
      END IF
  100 READ(10,'(A6,A74)',END=1000)L1,L2
      L1=UPPERC(L1)
      L2=UPPERC(L2)
      DO 10 Z=1,5
      IF (SUBSTR(L1,Z,1).NE.' ') THEN
         IF (SUBSTR(L1,Z,1).EQ.'!') THEN
            DOCE=1
            CALL PRIMI(L2,L1,Z)
            GO TO 100
         ELSE IF(SUBSTR(L1,Z,1).NE.'!') THEN
                DO 5 Y=1,74
    5           IF (SUBSTR(L2,Y,1).NE.' ') GO TO 6
                GO TO 100
    6           CAR=SUBSTR(L1,Z,1)
                X=ICHAR(CAR)
                IF(X.GE.48.AND.X.LE.57) THEN
                  WRITE(11,'(A6,A)')L1,SUSBTR(L2,Y,74-Y)
                  GO TO 100
                END IF
                WRITE(11,'(''C'',A,A)')SUBSTR(L1,Z+1,6-Z),
     *SUBSTR(L2,Y,74-Y)
                GO TO 100
              END IF
         END IF
   10 CONTINUE
C     PARTE PARA DETECTAR EL CARACTER DE CONTINUACION
      IF (SUBSTR(L1,6,1).NE.' ') THEN
         DO 7 Y=1,74
    7    IF(SUBSTR(L2,Y,1).NE.' ')GO TO 8
         GO TO 100
    8    WRITE(11,'(5X,''*'',A)')SUBSTR(L2,Y,74-Y)
         GO TO 100
      END IF
      DO 11 I=1,74
   11 IF(SUBSTR(L2,I,1).NE.' ')GO TO 20
C     EL PROGRAMA DE ENTRADA PUEDE TENER SANGRIA
      GO TO 100
C     PUEDE EXISTIR LINEA EN BLANCO
```

```
   20 IF(SUBSTR(L2,I,5).EQ.'WHILE')THEN
                CALL WHILE(L1,L2,I)
                GO TO 100
      END IF
      IF(SUBSTR(L2,I,6).EQ.'REPEAT')THEN
                CALL REPEAT(L1,L2)
                GO TO 100
      EN DIF
      IF(SUBSTR(L2,I,4).EQ.'FOR ')THEN
    3           CALL FOR(L1,L2,I)
                GO TO 100
      END IF
      IF(SUBSTR(L2,I,4).EQ.'CASE')THEN
                CALL CASE(L1,L2,I)
                GO TO 100
      EN DIF
      IF(SUBSTR(L2,I,5).EQ.'ARRAY'.OR.SUBSTR(L2,I,3).EQ.'VAR')THEN
                CALL VAR(L1,L2)
                GO TO 100
      END IF
      WRITE(11,'(A6,A)')L1,SUBSTR(L2,I,74-I)
C     COPIA LA LINEA TAL Y COMO ESTA.
      GO TO 100
 1000 IF (DOCE.EQ.1) THEN
        REWIND 12
        DO 30 J=1,300
        READ(12,'(A80)',END=1001)LN
   30   WRITE(11,'(A80)')LN
      END IF
 1001 IF(BITS(M,11,1).EQ.1)THEN
        WRITE(11,'(''@MAP'')')
        IF(ARCH2(1:3).NE.'    ')THEN
          I1=INDEX(ARCH2,'.')
          IF (SUBSTR(ARCH2,I1+1,1).EQ.' ')THEN
              SUBSTR(ARCH2,I1+1,6)='MATFOR'
          END IF
          IL=INDEX(ARCH2,' ')
          LN='@COPY,A TPF$.NAME$,'//ARCH2(1:IL-1)
          WRITE(11,'(A80)')LN
        END IF
        WRITE(11,'(''@BRKPT PRINT$'')')
        CLOSE(11)
        IERR=FACSF2('@ADD 11. . ')
        IF(IERR.NE.0)THEN
          IER=FACSF2('@ADD 11. . ')
          IF(IER.NE.0)THEN
             WRITE(6,'('' NO PUDE LANZAR EL ARCHIVO 11. '')')
             WRITE(6,'('' P.F. DA UN @ADD 11. '')')
          END IF
        END IF
      ELSE IF(ARCH2(1:3).NE.'    ')THEN
          CALL FACSF('@ASG,T '//FOA(1:IM-1)//'*SYM. . ')
          CALL FACSF('@USE 20.,'//FOA(1:IM-1)//'*SYM. . ')
          I1=INDEX(ARCH2,'.')
```

```
      IF  (SUBSTR(ARCH2,I1+1,1).EQ.' ')THEN
            SUBSTR(ARCH2,I1+1,6)='MATFOR'
      END IF
      IL=INDEX(ARCH2,' ')
      LN='@COPY,I '//FQA(1:IM-1)//'*MATFOR.,'//ARCH2(1:IL-1)
      WRITE(20,'(A80)')LN
      CLOSE(20)
      IERR=FACSF2('@ADD 20. . ')
      IF(IERR.NE.0)THEN
         IER=FACSF2('@ADD SYM. . ')
         IF(IER.NE.0)THEN
           WRITE(6,'('' NO PUDE LANZAR EL ARCHIVO 20. '')')
           WRITE(6,'('' P.F. DA UN @ADD 20. '')')
         END IF
      END IF
   END IF
   CALL FACSF('@FREE 10. . ')
   CALL FACSF('@FREE 12. . ')
   WRITE(6,'(1X,''*END MATFOR '')')
   STOP
   END
```

KBB*SECTES.ARCHIV

```
   SUBROUTINE ARCHIV(ARCH1,ARCH2,FQA)
   CHARACTER*80 ARCH1,ARCH2,ARCAX1,ARCAX2
   CHARACTER*12 FQ,FN,EN,EV,FC*6,RK*6,WK*6,EC*6
   CHARACTER*12 FQA,FNA,ENA,EVA,FCA*6,RKA*6,WKA*6,ECA*6
   CHARACTER*8 FECHA,TIME,MOMENT*20
   CALL ADATE(FECHA,TIME)
   MOMENT=FECHA(1:2)//'/'//FECHA(3:4)//'/'//FECHA(5:6)//'-'
  *//TIME(1:2)//':'//TIME(3:4)//':'//TIME(5:6)
   WRITE(6,'('' MATFOR SGC/RHR VER.01 '',A)')MOMENT
   CALL INFORM              @ . GENERA  TABLA RINF*
   CALL ARCH(1,FQ,FN,FC,RK,WK)
   NP=2
   ICAMPO=0
50 ICAMPO=ICAMPO+1
   CALL FFDASC(NP,FQ,FQA)
   IF(FQA.EQ.'@@@@@@@@@@@@')THEN
     CALL CALIF(FQ)         @ . OBTIENE EL CALIFICADOR DEL
     NP=2                     . DEL USUARIO
     CALL FFDASC(NP,FQ,FQA)
   END IF
   NP=2
   CALL FFDASC(NP,FN,FNA)
   NP=1
   CALL FFDASC(NP,FC,FCA)
   NP=1
   CALL FFDASC(NP,RK,RKA)
   NP=1
   CALL FFDASC(NP,WK,WKA)
```

```
NP=2
CALL FFDASC(NP,EN,ENA)
NP=2
CALL FFDASC(NP,EV,EVA)
NP=1
CALL FFDASC(NP,EC,ECA)
I1=INDEX(FQA,' ')
IF(I1.EQ.0)I1=13
ARCAX1=FQA(1:I1-1)//'*'
IF(FNA(1:2).EQ.'@@')THEN
  IF(ICAMPO.EQ.1)GO TO 1000
  GO TO 100
END IF
ARCAX2=ARCAX1
IL=INDEX(ARCAX2,' ')
I2=INDEX(FNA,' ')
IF(I2.EQ.0)I2=13
ARCAX1=ARCAX2(1:IL-1)//FNA(1:I2-1)
IF(FCA(1:2).EQ.'@@')FCA='1'
ARCAX2=ARCAX1
IL=INDEX(ARCAX2,' ')
I3=INDEX(FCA,' ')
IF(I3.EQ.0)I3=7
ARCAX1=ARCAX2(1:IL-1)//'('//FCA(1:I3-1)//')'
IF(RKA(1:2).NE.'@@')THEN
  ARCAX2=ARCAX1
  IL=INDEX(ARCAX2,' ')
  I4=INDEX(RKA,' ')
  IF(I4.EQ.0)I4=7
  ARCAX1=ARCAX2(1:IL-1)//'/'//RKA(1:I4-1)
  IF(WKA(1:2).NE.'@@')THEN
    ARCAX2=ARCAX1
    IL=INDEX(ARCAX2,' ')
    I5=INDEX(WKA,' ')
    IF(I5.EQ.0)I5=7
    ARCAX1=ARCAX2(1:IL-1)//'/'//WKA(1:I5-1)
  END IF
ELSE IF(WKA(1:2).NE.'@@')THEN
    ARCAX2=ARCAX1
    IL=INDEX(ARCAX2,' ')
    I5=INDEX(WKA,' ')
    IF(I5.EQ.0)I5=7
    ARCAX1=ARCAX2(1:IL-1)//'//'//WKA(1:I5-1)
END IF
ARCAX2=ARCAX1
IL=INDEX(ARCAX2,' ')
ARCAX1=ARCAX2(1:IL-1)//'.'
IF(ICAMPO.EQ.1)THEN
  ARCH1=ARCAX1
  GO TO 40
END IF
GO TO 60
```

```
 40 FQA='
    FNA='
    FCA='
    RKA='
    WKA='
    ARCAX1='
    ARCAX2='
    NP=3
    CALL ELT(2,FQ,FN,FC,RK,WK,EN,EV,EC)
    GO TO 50
 60 IF(ENA(1:2).EQ.'@@')THEN
      ARCH2=ARCAX1
      RETURN
    END IF
    ARCAX2=ARCAX1
    IL=INDEX(ARCAX2,' ')
    I6=INDEX(ENA,' ')
    IF(I6.EQ.0)I6=13
    ARCAX1=ARCAX2(1:IL-1)//ENA(1:I6-1)
    IF(EVA(1:2).NE.'@@')THEN
      ARCAX2=ARCAX1
      IL=INDEX(ARCAX2,' ')
      I7=INDEX(EVA,' ')
      IF(I7.EQ.0)I7=13
      ARCAX1=ARCAX2(1:IL-1)//'/'//EVA(1:I7-1)
      IF(ECA(1:2).NE.'@@')THEN
        ARCAX2=ARCAX1
        IL=INDEX(ARCAX2,' ')
        I8=INDEX(ECA,' ')
        IF(I8.EQ.0)I8=7
        ARCAX1=ARCAX2(1:IL-1)//'('//ECA(1:I8-1)//')'
      END IF
    ELSE IF(ECA(1:2).NE.'@@')THEN
        ARCAX2=ARCAX1
        IL=INDEX(ARCAX2,' ')
        I8=INDEX(ECA,' ')
        IF(I8.EQ.0)I8=7
        ARCAX1=ARCAX2(1:IL-1)//'('//ECA(1:I8-1)//')'
    END IF
    ARCH2=ARCAX1
    RETURN
100 ARCH2='
    RETURN
1000 WRITE(6,'('' ERROR!!... EN EL LLAMADO DEL PRE-PROCESADOR
    $'')')
    WRITE(6,'('' (FALTO EL PUNTO EN EL ARCHIVO DE ENTRADA)'')')
    STOP
    END
```

KBB4SBCTES. INFORM

AXR$

```
. RUTINA CREADA CON EL FIN DE OBTENER LA INFORMACION                    .
. PROPORCIONADA  POR EL PROCEDIMIENTO ELT$.                             .
. PARA OBTENER LOS PARAMETROS MANDADOS POR EL PROGRAMADOR:              .
. COMO EL ARCHIVO DE ENTRADA Y EL ARCHIVO.ELEMENTO                      .
. (ARCHIVO PUNTO ELEMENTO)  DE SALIDA                                   .
. FORMA DE USO:                                                         .
. CALL INFORM          @ SOLO UNA VEZ EN LA CORRIDA                     .
. CALL ARCH(N,FQUAL,FNAME,FCYC,RKEY,WKEY)                               .
. CALL ELT(N,FQUAL,FNAME,FCYC,RKEY,WKEY,ENAME,EVER,ECYC)                .
. EJ:                                                                   .
. CHARACTER*12 FQUAL,FNAME,ENAME,EVER,FCYC*6,RKEY*6,WKEY*6,ECYC*6       .
. CALL INFORM                                                           .
. CALL ARCH(N,FQUAL,FNAME,FCYC,RKEY,WKEY)                               .
. LOGICA PROGRAMACIONAL                                                 .
. CALL ELT(N,FQUAL,FNAME,FCYC,RKEY,WKEY,ENAME,EVER,ECYC)                .
. DONDE N= NUMERO DEL CAMPO DESEADO                                     .
. DONDE FQUAL= CALIFICADOR DEL ARCHIVO                                  .
. DONDE FNAME= NOMBRE DEL ARCHIVO                                       .
. DONDE FCYC= CICLO DEL ARCHIVO                                         .
. DONDE RKEY= LLAVE DE LECTURA                                          .
. DONDE WKEY= LLAVE DE ESCRITURA                                        .
. DONDE ENAME= NOMBRE DEL ELEMENTO                                      .
. DONDE EVER = VERSION DEL ELEMENTO                                     .
. DONDE ECYC = CICLO DEL ELEMENTO                                       .
. EL UNICO PARAMETRO QUE PASA FORTRAN ES N                              .
```

```
ELT$
F2466       FORM        24,6,6
PF          FORM        12,6,18
FORM3       FORM        6,3,9,18
BUFF$       RES         100
SALVA       RES         5
MSG ' ERROR EN EL LLAMADO DEL PRE-PROCESADOR *MATFOR* DE SGC/RHR
VER.01'
LM $EQU $-MSG
INFORM$ .
 .          slj         ton$                .
            S           A0,SALVA            . SALVA A0
            S           X11,SALVA+1         . SALVA X11
            LXM,U       X1,0                . U DIR INMEDIATO
            LXI,U       X1,1                .
            L,U         R1,99               .
            L           A0,('       ')      .
OTRA        S           A0,BUFF,*X1         . ES UN LOOP Y
            JGD         R1,OTRA             . GUARDA BLANCOS EN BUFF
            SZ          INFOR$              . GUARDA CEROS EN INFOR$
            L           A0,(100,BUFF)       .
```

```
         LMJ      X11,RINF$          . CARGA EN X11 <--FIN Y
                                     . SALTA A LA
                                     . TABLA RINF$ Y OBTIENE LA
                                     . INFORMACION DEL CAMPO
                                     . DESEADO
         J        FIN                . REGRESO ANORMAL J 0,X11
         L        X11,SALVA+1        . REGRESO NORMAL J 1,X11
                                     . REST X11
         J        0,X11              . REGRESA
ARCH$ .
 .       SLJ      TON$               . ES UN DEBUG
         S        A0,SALVA           . SALVA A0
         S        X11,SALVA+1        . SALVA X11
         L        A1,$0,A0           . OBTIENE PRIMER PARAMETRO
         L        A2,(F2466 0,01,0)  . FORMA PARA ELT$
         S        A2,SALVA+2         . LO SALVA
         S,S6     A1,SALVA+2         . SALVA NUMERO DE CAMPO
         L        A0,SALVA+2         . TOMA PRIMER PARAMETRO
         LMJ      X11,SELT$          . OBTIENE TABLA ELT$
         J        FIN                . REGRESO ANORMAL
         L        A0,SALVA           . RESTAURA A0
         DL       A1,FQUAL           . TOMA CALIFICADOR (2
                                     . PARAMETRO)
         DS       A1,$1,A0           . PASA  CALIFICADOR
         DL       A1,FNAME           . TOMA NOMBRE ARCHIVO (3
                                     . PARAMETRO)
         DS       A1,$2,A0           . PASA NOMBRE ARCHIVO
         L        A1,FCYC            . TOMA CICLO DEL ARCHIVO (4
                                     . PARAMETRO)
         S        A1,$3,A0           . PASA CICLO ARCHIVO
         L        A1,RKEY            . TOMA LLAVE DE LECTURA (5
                                     . PARAMETRO)
         S        A1,$4,A0           . PASA LLAVE DE LECTURA
         L        A1,WKEY            . TOMA LLAVE DE ESCRITURA (6
                                     . PARAMETRO)
         S        A1,$5,A0           . PASA LLAVE DE ESCRITURA
         L        A0,SALVA           . RESTAURA A0
         L        X11,SALVA+1        . RESTAURA X11
         J        0,X11              .
 ..........................................................
 . parte para obtener el segundo campo, si es que lo manda el   .
 . usuario y si no regresar sin error                           .
 ..........................................................
ELT$ .
 .       SLJ      TON$               .
         S        A0,SALVA           .
         S        X11,SALVA+1        .
         L        A1,$0,A0           . OBTIENE PRIMER PARAMETRO
         L        A2,(F2466 0,01,0)  . FORMA PARA ELT$
         S        A2,SALVA+2         . LO SALVA
         S,S6     A1,SALVA+2         . SALVA NUMERO DE CAMPO
                                     . (SEGUNDO)
```

```
         L        A0,SALVA+2          . TOMA PRIMER PARAMETRO
         LMJ      X11,SELT$           . OBTIENE TABLA ELT$
         J        FINN                . REGRESO ANORMAL
         L        A0,SALVA            . RESTAURA A0
         DL       A1,FQUAL            . TOMA CALIFICADOR (2
                                      . PARAMETRO)
         DS       A1,*1,A0            . PASA  CALIFICADOR
         DL       A1,FNAME            .
         DS       A1,*2,A0            .
         L        A1,FCYC             .
         S        A1,*3,A0            .
         L        A1,RKEY             .
         S        A1,*4,A0            .
         L        A1,WKEY             .
         S        A1,*5,A0            .
         DL       A1,ENAME            . TOMA NOMBRE DEL ELEMENTO
                                      . (7 PARAM)
         DS       A1,*6,A0            . PASA EL NOMBRE DEL
                                      . ELEMENTO
         DL       A1,EVER             . TOMA VERSION DEL ELEMENTO
                                      . (8 PARAM)
         DS       A1,*7,A0            . PASA VERSION DEL ELEMENTO
         L        A1,ECYC             . TOMA CICLO DEL ELEMENTO
                                      . (9 PARAM)
         S        A1,*8,A0            . PASA CICLO DEL ELEMENTO
         L        A0,SALVA            . RESTAURA A0
         L        X11,SALVA+1         . RESTAURA X11
         J        0,X11               .
FINN     L        X11,SALVA+1         .
         J        0,X11               .
FIN      LA       A0,(PF 1,LM,MSG)    .
         ER       PRINT$              .
         ER       EXIT$               .
         END      .                   .
```

KB0:SBCTES.CALIF

```
                    AXR$
 .....................................................................
 . OBTIENE EL CALIFICADOR DE LA PCT Y LO PASA A FORTRAN              .
 . FORM DE USO$                                                      .
 . CALL CALIF(Q)                                                     .
 . DONDE Q ES CHARACTER DE  12  (LO PASA EN FIELDATA)                .
 .....................................................................
SALVA     RES       5
CALIF$
          S         A0,SALVA          . SALVA A0
          S         X11,SALVA+1       . SALVA X11
          L,U       A0,SALVA+2        . DIRECCION PARA EL CALIF.
          L         A1,(2,266)        . DOS PALABRAS
          ER        PCT$              . OBTIENE CALIF.
          L         A0,SALVA          . RESTAURA A0
          DL        A1,SALVA+2        . TOMA CALIF.
          DS        A1,$0,A0          . PASA CALIF. A FTN
          L         X11,SALVA+1       . RESTAURA X11
          J         0,X11             . REGRESA A FTN
          $END
```

KB0:SBCTES.TABINI

```
      SUBROUTINE TABINI
      COMMON /STATUS/STA
      COMMON /VARBLE/VARS,ETI
      CHARACTER STA$6(22,2),VARS$6(100,5)
C INICIALIZACION DE LA TABLA DE VARIABLES E
C INICIALIZACION DE LA TABLA DE STATUS
      DO 1 I=1,100
      DO 1 J=1,5
    1 VARS(I,J)='      '
      DATA (STA(I,1),I=1,11)/'SUMVEC','RESVEC','MULVEC','MVMVEC',
     *'MVEVEC','MEVVEC','MAXVEC','MINVEC','MEDVEC',
     *'SUMMAT','RESMAT'/
      DATA (STA(I,1),I=12,22)/'MULMAT','MMVMAT','MMEMAT','MEMMAT',
     *'INVMAT','IMPVEC','IMVVEC','LEEVEC','LEVVEC',
     *'IMPMAT','LEEMAT'/
      DATA (STA(I,2),I=1,22)/22*'0'/
      RETURN
      END
```

KBD$SBCTES.OPT

```
                    AXR$
.....................................................................
.   RUTINA CREADA CON EL FIN DE PASARLE A FORTRAN EN UNA PALABRA.    .
.   LAS OPCIONES CON LA CUALES FUE EJECUTADO UN PROGRAMA.           .
.   FORMA DE LLAMAR:                                                .
.   CALL OPT(N)                                                     .
.   EN N QUEDAN LAS OPCIONES EN LA NOTACION DE:                     .
.   BIT 25 ENCENDIDO SIGNIFICA OPCION A                             .
.   BIT 24 ENCENDIDO SIGNIFICA OPCION B ETC.                        .
.....................................................................
SALVA     RES       3
OPT$.
          SA        A0,SALVA          . SALVA A0
          S         X11,SALVA+1       . SALVA X11
          ER        OPT$              . OPTIENE OPCIONES EN A0
          S         A0,SALVA+2        . LAS SALVA
          L         A2,SALVA+2        . LAS TOMA EN A2
          L         A0,SALVA          . RESTAURA A0
          S         A2,*0,A0          . PASA OPCIONES A FTN
          L         X11,SALVA+1       . RESTAURA X11
          J         0,X11             . REGRESA
          $END
```

* **PARTE 2:** Programas para las estructuras de control.

[illegible]SBCTES.WHILE

```
      SUBROUTINE WHILE(L1,L2,I)
C     SUBRUTINA PARA EL LLAMADO DE LA ESTRUCTURA "WHILE"
      IMPLICIT INTEGER(A-Z)
      COMMON /ETIKET/ETI1,ETI2,DOCE
      DIMENSION IC(5)
      CHARACTER L1*6,L2*74,CAR*1
      IL=0
      IOP=0
      ICL=0
C     IL=LONG. DE LA CONDICION--IOP=DONDE ABRE EL PRIMER
C     PARENTESIS
C     ICL=DONDE CIERRA EL ULTIMO PARENTESIS
      I=I+5
      IF(SUBSTR(L2,I,1).NE.' ')THEN
        WRITE(6,'(' ERROR!!...FALTO ESPACIO ENTRE EL WHILE Y LA
     *CONDICION..."WHILE (CONDI.)" '')')
      STOP
      END IF
      DO 50 J=I,74
   50 IF(SUBSTR(L2,J,1).NE.' ')GO TO 52
      STOP ' ERROR!!... FALTO CONDICION EN EL "WHILE" '
   52 I=J
      DO 15 J=I,74
         IF(SUBSTR(L2,J,2).EQ.'DO')GO TO 20
         IL=IL+1
         IF(SUBSTR(L2,J,1).EQ.'(')THEN
                  IOP=IOP+1
                  IC(IOP)=J
                  GO TO 15
         ELSE IF(SUBSTR(L2,J,1).EQ.')')THEN
                       ICL=ICL+1
                       GO TO 15
         END IF
   15 CONTINUE
      STOP ' ERROR!!... MALA DECLARACION DEL "WHILE" FALTO "DO" '
   20 IF(SUBSTR(L2,J-1,1).NE.' ')THEN
        STOP ' ERROR!!... FALTO ESPACIO ENTRE "CONDICION" Y "DO" '
      END IF
      IF(IOP.EQ.0)THEN
        STOP ' ERROR!!... CONDICION DEL "WHILE" VA ENTRE PARENTE
     *SIS'
      END IF
      IF(IOP.NE.ICL)THEN
        STOP ' ERROR!!... PARENTESIS INCOMPLETOS EN EL "WHILE" '
      END IF
```

```
      IOP=IC(1)
      ETI1=ETI2+1
      ETI2=ETI1+1
      NWAX1=ETI1
      NWAX2=ETI2
      WRITE(11,'(I5,1X,''IF (.NOT.'',A,'') GO TO'',1X,I5)')NWAX1,
     *SUBSTR(L2,IOP,IL),NWAX2
      JC=0
  101 READ(10,'(A6,A74)',END=102)L1,L2
      L1=UPPERC(L1)
      L2=UPPERC(L2)
      DO 10 Z=1,5
      IF(SUBSTR(L1,Z,1).NE.' ') THEN
        IF(SUBSTR(L1,Z,1).EQ.'!') THEN
          DOCE=1
          CALL PRIMI(L2,L1,Z)
          GO TO 101
        ELSE IF(SUBSTR(L1,Z,1).NE.'!')THEN
               DO 5 Y=1,74
     5         IF(SUBSTR(L2,Y,1).NE.' ')GO TO 6
               GO TO 101
     6         CAR=SUBSTR(L1,Z,1)
               X=ICHAR(CAR)
               IF(X.GE.48.AND.X.LE.57)THEN
                 WRITE(11,'(A6,A)')L1,SUBSTR(L2,Y,74-Y)
                 GO TO 101
               END IF
               WRITE(11,'(''C'',A,A)')SUBSTR(L1,Z+1,6-Z),
     *SUBSTR(L2,Y,74-Y)
               GO TO 101
        END IF
      END IF
   10 CONTINUE
C     PARTE PARA DETECTAR EL CARACTER DE CONTINUACION...
      IF(SUBSTR(L1,6,1).NE.' ')THEN
        DO 7 Y=1,74
    7   IF(SUBSTR(L2,Y,1).NE.' ')GO TO 8
        GO TO 101
    8   WRITE(11,'(5X,''*'',A)')SUBSTR(L2,Y,74-Y)
        GO TO 101
      END IF
      JC=JC+1
      DO 17 I=1,74
   17   IF(SUBSTR(L2,I,1).NE.' ')GO TO 22
      GO TO 101
C     PUEDE EXISTIR LINEA EN BLANCO
   22 IF(SUBSTR(L2,I,5).EQ.'BEGIN')GO TO 101
      IF(SUBSTR(L2,I,6).EQ.'REPEAT')THEN
               CALL REPEAT(L1,L2)
               GO TO 101
      END IF
```

```
      IF(SUBSTR(L2,I,4).EQ.'FOR ')THEN
                CALL FOR(L1,L2,I)
                GO TO 101
      END IF
      IF(SUBSTR(L2,I,5).EQ.'CASE')THEN
                CALL CASE(L1,L2,I)
                GO TO 101
      END IF
      IF(SUBSTR(L2,I,5).EQ.'WHILE')GO TO 102
      IF(SUBSTR(L2,I,4).EQ.'ENDW'.OR.SUBSTR(L2,I,5).EQ.'END W')
     *GO TO 18
      WRITE(11,'(A6,A)')L1,SUBSTR(L2,I,74-I)
      IF(JC.GT.1)GO TO 101
   18 WRITE(11,'(6X,''GO TO'',1X,I5)')NWAX1
      WRITE(11,'(I5,1X,''CONTINUE'')')NWAX2
      RETURN
  102 STOP ' ERROR!!... FALTO "ENDW" O "END WHILE" '
      END
```

KBB#SBCTEB.REPEAT

```
      SUBROUTINE REPEAT(L1,L2)
C     SUBRUTINA PARA EL LLAMADO DE LA ESTRUCTURA "REPEAT-UNTIL"
      IMPLICIT INTEGER(A-Z)
      COMMON /ETIKET/ETI1,ETI2,DOCE
      DIMENSION IC(5)
      CHARACTER L1*6,L2*74,CAR*1
      IL=0
      IOP=0
      ICL=0
      ETI2=ETI2+1
      NRAX=ETI2
      WRITE(11,'(I5,1X,''CONTINUE'')')NRAX
  105 READ(10,'(A6,A74)',END=106)L1,L2
      L1=UPPERC(L1)
      L2=UPPERC(L2)
      DO 10 Z=1,5
      IF(SUBSTR(L1,Z,1).NE.' ') THEN
        IF(SUBSTR(L1,Z,1).EQ.'!') THEN
          DOCE=1
          CALL PRIMI(L2,L1,Z)
          GO TO 105
        ELSE IF(SUBSTR(L1,Z,1).NE.'!')THEN
              DO 5 Y=1,74
    5         IF(SUBSTR(L2,Y,1).NE.' ')GO TO 6
              GO TO 105
    6         CAR=SUBSTR(L1,Z,1)
              X=ICHAR(CAR)
              IF(X.GE.48.AND.X.LE.57)THEN
                WRITE(11,'(A6,A)')L1,SUBSTR(L2,Y,74-Y)
                GO TO 105
              END IF
```

```
              WRITE(11,'(''C'',A,A)')SUBSTR(L1,Z+1,6-Z),
     $SUBSTR(L2,Y,74-Y)
              GO TO 105
        END IF
      END IF
   10 CONTINUE
C     PARTE PARA DETECTAR EL CARACTER DE CONTINUACION...
      IF(SUBSTR(L1,6,1).NE.' ')THEN
        DO 7 Y=1,74
    7   IF(SUBSTR(L2,Y,1).NE.' ')GO TO 8
        GO TO 105
    8   WRITE(11,'(5X,''$'',A)')SUBSTR(L2,Y,74-Y)
        GO TO 105
      END IF
      DO 30 I=1,74
   30    IF(SUBSTR(L2,I,1).NE.' ')GO TO 40
      GO TO 105
   40 IF(SUBSTR(L2,I,5).EQ.'UNTIL'.OR.SUBSTR(L2,I,4).EQ.'UNT')
     *THEN
        I=I+4
C     IF(SUBSTR(L2,I,5).EQ.'UNTIL')I=I+5
        DO 35 J=I,74
           IL=IL+1
           IF(SUBSTR(L2,J,1).EQ.'(')THEN
                    IOP=IOP+1
                    IC(IOP)=J
                    GO TO 35
           ELSE IF(SUBSTR(L2,J,1).EQ.')')THEN
                    ICL=ICL+1
                    IF(IOP.EQ.ICL)GO TO 41
                    GO TO 35
           END IF
   35    CONTINUE
         IF(IOP.NE.0)THEN
         STOP ' ERROR!!... PARENTESIS INCOMPLETOS EN EL "REPEAT-
     $UNTIL" '
         END IF
         STOP ' ERROR!!... FALTO CONDICION DEL "REPEAT-UNTIL"
   41    IOP=IC(1)
         WRITE(11,'(6X,''IF(.NOT.'',A,'')GO TO'',I5)')
     $SUBSTR(L2,IOP,IL),NRAX
         RETURN
         END IF
         IF(SUBSTR(L2,I,5).EQ.'WHILE')THEN
                  CALL WHILE(L1,L2,I)
                  GO TO 105
         END IF
         IF(SUBSTR(L2,I,4).EQ.'FOR ')THEN
                  CALL FOR(L1,L2,I)
                  GO TO 105
         END IF
```

```
      IF(SUBSTR(L2,I,4).EQ.'CASE')THEN
                CALL CASE(L1,L2,I)
                GO TO 105
      END IF
      IF(SUBSTR(L2,I,6).EQ.'REPEAT')GO TO 106
      WRITE(11,'(A6,A)')L1,SUBSTR(L2,I,74-I)
      GO TO 105
  106 STOP ' ERROR!!... NO SE ENCONTRO "UNTIL" DEL "REPEAT" '
      END
```

KBBSECTES.FOR

```
      SUBROUTINE FOR(L1,L2,I)
C     SUBRUTINA PARA EL LLAMADO DE LA ESTRUCTURA "FOR"
      IMPLICIT INTEGER(A-Z)
      COMMON /ETIKET/ETI1,ETI2,DOCE
      CHARACTER L1*6,L2*74,CAR*1
      JJ=0
      IT=0
      IT1=0
      IT2=0
C     II=INICIO DE LA CADENA--IT=TERMINACION DE LA CADENA
      ETI2=ETI2+1
      NF2=ETI2
      II=I+4
      DO 52 J=II,74
         IF(SUBSTR(L2,J,1).EQ.'I'.OR.SUBSTR(L2,J,1).EQ.'=')
     *GO TO 54
   52 IT=IT+1
      STOP ' ERROR!!... MALA DECLARACION PARA EL "FOR" '
   54 IF(IT.GT.6)THEN
        STOP 'ERROR!!... VARIABLE MAYOR DE 6 DIGITOS EN EL "FOR"'
      END IF
      J=J+1
      IF(SUBSTR(L2,J,1).EQ.'=')J=J+1
      II=J
      DO 58 J=J,74
         IF(SUBSTR(L2,J,1).EQ.'T'.OR.SUBSTR(L2,J,1).EQ.'D')
     *GO TO 60
   58 IT1=IT1+1
   60 J=J+2
      IF(SUBSTR(L2,J,1).EQ.'W')THEN
               J=J+4
             JJ=JJ+1
      END IF
      II2=J
      DO 65 J=J,74
         IF(SUBSTR(L2,J,4).EQ.'DO  ')GO TO 67
   65 IT2=IT2+1
      STOP ' ERROR!!... DECLARACION DEL "FOR" INCOMPLETA FALTO
     *"DO" '
```

```
 67 IF(SUBSTR(L2,J-1,1).EQ.' ')GO TO 68
    STOP ' ERROR!!... FALTO ESPACIO PARA EL "DO" EN EL "FOR"
 68 IF(JJ.GE.1)THEN
       WRITE(11,'(A6,''DO'',1X,I5,1X,A,''='',A,'',1,-'',A)')
   $L1,NF2,$SUBSTR(L2,II,IT),SUBSTR(L2,III,(IT1-1)),
    $SUBSTR(L2,II2,(IT2-1))
       JC=0
       GO TO 103
    END IF
    WRITE(11,'(A6,''DO '',I5,1X,A,''='',A,'','',A)')L1,NF2,
   $SUBSTR(L2,II,IT),SUBSTR(L2,III,(IT1-1)),
   $SUBSTR(L2,II2,(IT2-1))
    JC=0
103 READ(10,'(A6,A74)',END=104)L1,L2
    L1=UPPERC(L1)
    L2=UPPERC(L2)
    DO 10 Z=1,5
    IF(SUBSTR(L1,Z,1).NE.' ') THEN
      IF(SUBSTR(L1,Z,1).EQ.'!') THEN
        DOCE=1
        CALL PRIMI(L2,L1,Z)
        GO TO 103
      ELSE IF(SUBSTR(L1,Z,1).NE.'!')THEN
             DO 5 Y=1,74
  5          IF(SUBSTR(L2,Y,1).NE.' ')GO TO 6
             GO TO 103
  6          CAR=SUBSTR(L1,Z,1)
             X=ICHAR(CAR)
             IF(X.GE.48.AND.X.LE.57)THEN
               WRITE(11,'(A6,A)')L1,SUBSTR(L2,Y,74-Y)
               GO TO 103
             END IF
             WRITE(11,'(''C'',A,A)')SUBSTR(L1,Z+1,6-Z),
   $SUBSTR(L2,Y,74-Y)
             GO TO 103
      END IF
    END IF
 10 CONTINUE
C   PARTE PARA DETECTAR DEL CARACTER DE CONTINUACION...
    IF(SUBSTR(L1,6,1).NE.' ')THEN
      DO 7 Y=1,74
  7   IF(SUBSTR(L2,Y,1).NE.' ')GO TO 8
      GO TO 103
  8   WRITE(11,'(5X,''$'',A)')SUBSTR(L2,Y,74-Y)
      GO TO 103
    END IF
    JC=JC+1
```

```
      DO 70 IN=1,74
 70     IF(SUBSTR(L2,IN,1).NE.' ')GO TO 75
      GO TO 103
 75   IF(SUBSTR(L2,IN,5).EQ.'BEGIN')GO TO 103
      IF(SUBSTR(L2,IN,6).EQ.'REPEAT')THEN
                CALL REPEAT(L1,L2)
                NF1=NF2
                GO TO 103
      END IF
      IF(SUBSTR(L2,IN,5).EQ.'WHILE')THEN
                CALL WHILE(L1,L2,IN)
                NF1=NF2
                GO TO 103
      END IF
      IF(SUBSTR(L2,IN,4).EQ.'CASE')THEN
                CALL CASE(L1,L2,IN)
                GO TO 103
      END IF
      IF(SUBSTR(L2,IN,4).EQ.'FOR ')GO TO 104
      IF(SUBSTR(L2,IN,4).EQ.'ENDF'.OR.SUBSTR(L2,IN,5).EQ.'END F')
     #GOTO 80
      WRITE(11,'(A6,A)')L1,SUBSTR(L2,IN,74-IN)
      IF(JC.GT.1)GO TO 103
 80   WRITE(11,'(I5,1X,''CONTINUE'')')NF2
      RETURN
104   STOP ' ERROR!!... NO SE ENCONTRO "ENDF" O "END FOR" '
      END
```

KBBISBCTES.CASE

```
      SUBROUTINE CASE(L1,L2,I)
C     SUBRUTINA PARA EL LLAMADO DE LA ESTRUCTURA "CASE"
      IMPLICIT INTEGER(A-Z)
      COMMON /ETIKET/ETI1,ETI2,DOCE
      CHARACTER L1*6,L2*74,V*40(10),VAS*74,VC*40,CAR*1
      CIF=0
      BAN=0
      PA=0
      PC=0
      I=I+4
      IF(SUBSTR(L2,I,1).EQ.' ')GO TO 1
      STOP 'ERROR!!... MALA DECLARACION DEL "CASE" FALTO ESPACIO'
    1 DO 2 J=I,74
    2 IF(SUBSTR(L2,J,1).NE.' ')GO TO 4
      STOP 'ERROR!!... MALA DECLARACION DEL "CASE", *VARIABLE EN
     *CUESTION?? '
    4 M=J
      DO 6 J=1,40
      IF(SUBSTR(L2,M+J,1).EQ.' ')GO TO 8
      IF(SUBSTR(L2,M+J,1).EQ.'(')THEN
       PA=PA+1
       BAN=BAN+1
       IF(BAN.EQ.1)THEN
         IF(J.GT.6)THEN
         STOP 'ERROR!!... VARIABLE EN CUESTION DEL "CASE" MAYOR DE
     *6 DIGITOS '
         END IF
       END IF
      END IF
      IF(SUBSTR(L2,M+J,1).EQ.')')PC=PC+1
    6 CONTINUE
    8 IF(SUBSTR(L2,M+J-2,2).EQ.'OF')THEN
       STOP 'ERROR!!... FALTO ESPACIO PARA EL "OF" EN EL "CASE"'
      END IF
      IF(PA.EQ.0)THEN
       IF(J.GT.6)THEN
         STOP 'ERROR!!... VARIABLE EN CUESTION DEL "CASE" MAYOR DE
     *6 DIGITOS '
       END IF
      END IF
      IF(PA.NE.PC)THEN
       STOP 'ERROR!!... PARENTESIS INCOMPLETOS EN EL "CASE" '
      END IF
      VC=SUBSTR(L2,M,J)
      M=M+J
      DO 10 J=1,10
   10 IF(SUBSTR(L2,M+J,2).EQ.'OF')GO TO 100
      STOP 'ERROR!!... MALA DECLARACION DEL "CASE" FALTO "OF" '
  100 READ(10,'(A6,A74)',END=105)L1,L2
      L1=UPPERC(L1)
      L2=UPPERC(L2)
```

```
      DO 60 W=1,5
      IF(SUBSTR(L1,W,1).NE.' ')THEN
         DO 61 Y=1,74
   61    IF(SUBSTR(L2,Y,1).NE.' ')GO TO 62
         GO TO 100
   62 WRITE(11,'(''C'',A,A)')SUBSTR(L1,W+1,6-W),SUBSTR(L2,Y,74-Y)
         GO TO 100
      END IF
   60 CONTINUE
      CB=0
      L=0
      J=1
   15 DO 20 K=J,74
   20 IF(SUBSTR(L2,K,1).NE.' ')GO TO 21
      GO TO 100
   21 IF(SUBSTR(L2,K,5).EQ.'ENDC '.OR.SUBSTR(L2,K,5).EQ.'END C')
     *GOTO 150
         DO 22 I=1,40
         IF(SUBSTR(L2,K+I,1).EQ.',')THEN
          L=L+1
          V(L)=SUBSTR(L2,K,I)
          J=K+I+1
          GO TO 15
         ELSE IF(SUBSTR(L2,K+I,1).EQ.':')THEN
          L=L+1
          V(L)=SUBSTR(L2,K,I)
          J=K+I+1
          VAS=SUBSTR(L2,J,74-J)
   39     DO 40 Z=1,74
   40     IF(SUBSTR(VAS,Z,1).NE.' ')GO TO 44
C  PARTE DE ASIGNAMIENTO DEL CASE NO EXISTE EN ESTA LINEA
   90     READ(10,'(6X,A74)')L2
          L2=UPPERC(L2)
          DO 441 Z=1,74
          IF(SUBSTR(L2,Z,1).EQ.':')THEN
          STOP 'ERROR!!... FALTO DECLARACION EJECUTABLE DESPUES DE
     *LOS DOS PUNTOS'
          END IF
  441     CONTINUE
          DO 41 Z=1,74
   41     IF(SUBSTR(L2,Z,1).NE.' ')GO TO 42
          GO TO 90
   42     VAS=SUBSTR(L2,Z,74-Z)
          GO TO 39
   44     IF(SUBSTR(VAS,Z,5).EQ.'BEGIN')THEN
          CB=CB+1
          GO TO 110
   45     READ(10,'(A6,A74)',END=106)L1,L2
          L1=UPPERC(L1)
          L2=UPPERC(L2)
```

```
        DO 50 Z=1,5
        IF(SUBSTR(L1,Z,1).NE.' ') THEN
        IF(SUBSTR(L1,Z,1).EQ.'!') THEN
          DOCE=1
          CALL PRIMI(L2,L1,Z)
          GO TO 45
        ELSE IF(SUBSTR(L1,Z,1).NE.'!')THEN
              DO 51 Y=1,74
   51         IF(SUBSTR(L2,Y,1).NE.' ')GO TO 52
              GO TO 45
   52         CAR=SUBSTR(L1,Z,1)
              X=ICHAR(CAR)
              IF(X.GE.48.AND.X.LE.57)THEN
                WRITE(11,'(A6,A)')L1,SUBSTR(L2,Y,74-Y)
                GO TO 45
              END IF
              WRITE(11,'(''C'',A,A)')SUBSTR(L1,Z+1,6-Z),
     *SUBSTR(L2,Y,74-Y)
              GO TO 45
          END IF
        END IF
   50 CONTINUE
C       PARTE PARA EL CARACTER DE CONTINUACION...
        IF(SUBSTR(L1,6,1).NE.' ')THEN
          DO 55 Y=1,74
   55     IF(SUBSTR(L2,Y,1).NE.' ')GO TO 56
          GO TO 45
   56     WRITE(11,'(5X,''*'',A)')SUBSTR(L2,Y,74-Y)
          GO TO 45
        END IF
          DO 46 Y=1,74
   46     IF(SUBSTR(L2,Y,1).NE.' ')GO TO 48
          GO TO 45
   48     DO 93 YY=Y,68
          IF(SUBSTR(L2,YY,5).EQ.'BEGIN')THEN
            IF(SUBSTR(L2,YY-4,9).NE.'END BEGIN')GO TO 106
          END IF
   93     CONTINUE
          IF(SUBSTR(L2,Y,5).EQ.'WHILE')THEN
            CALL WHILE(L1,L2,Y)
            GO TO 45
          END IF
          IF(SUBSTR(L2,Y,6).EQ.'REPEAT')THEN
            CALL REPEAT(L1,L2)
            GO TO 45
          END IF
          IF(SUBSTR(L2,Y,3).EQ.'FOR')THEN
            CALL FOR(L1,L2,K)
            GO TO 45
          END IF
```

```
      IF(SUBSTR(L2,Y,5).EQ.'END B'.OR.SUBSTR(L2,Y,5).EQ.'ENDB')
     $GO TO 100
        VAS=SUBSTR(L2,Y,74-Y)
        DO 65 Z=1,74
        IF(SUBSTR(VAS,Z,1).NE.' ')THEN
          CB=CB+1
          IF(CB.GT.1)L=0
          GO TO 110
        END IF
 65     CONTINUE
      END IF
       GO TO 110
      END IF
 22   CONTINUE
      STOP 'ERROR!!... FALTARON LOS DOS PUNTOS (:) O "END CASE"'
110  X=INDEX(VC,' ')
     Z1=INDEX(V(1),' ')
     Z2=INDEX(V(2),' ')
     Z3=INDEX(V(3),' ')
     Z4=INDEX(V(4),' ')
     Z5=INDEX(V(5),' ')
     Z6=INDEX(V(6),' ')
     Z7=INDEX(V(7),' ')
     Z8=INDEX(V(8),' ')
     Z9=INDEX(V(9),' ')
     Z10=INDEX(V(10),' ')
     IF(X.EQ.0)X=41
     IF(Z1.EQ.0)Z1=41
     IF(Z2.EQ.0)Z2=41
     IF(Z3.EQ.0)Z3=41
     IF(Z4.EQ.0)Z4=41
     IF(Z5.EQ.0)Z5=41
     IF(Z6.EQ.0)Z6=41
     IF(Z7.EQ.0)Z7=41
     IF(Z8.EQ.0)Z8=41
     IF(Z9.EQ.0)Z9=41
     IF(Z10.EQ.0)Z10=41
     CIF=CIF+1
     IF(CIF.GE.2)GO TO 120
     IF(L.EQ.1)THEN
     WRITE(11,'(6X,''IF('',A,''.EQ.'',A,'')THEN'')')
    $VC(1:X-1),V(1)(1:Z1-1)
     ELSE IF(L.EQ.2)THEN
     WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,''
    $THEN'')')VC(1:X-1),V(1)(1:Z1-1),VC(1:X-1),V(2)(1:Z2-1)
     ELSE IF(L.EQ.3)THEN
     WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
    $,A,''.OR.'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),V(1)(1:Z1-1)
    $,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
```

```
 ELSE  IF(L.EQ.4)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),
*V(4)(1:Z4-1)
 ELSE  IF(L.EQ.5)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,'')
*THEN'')')VC(1:X-1),V(4)(1:Z4-1),VC(1:X-1),V(5)(1:Z5-1)
 ELSE  IF(L.EQ.6)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1),
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),V(4)(1:Z4-1)
*,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
 ELSE  IF(L.EQ.7)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1),
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
*,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),
*V(7)(1:Z7-1)
 ELSE  IF(L.EQ.8)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
*,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,'')
*THEN'')')VC(1:X-1),V(7)(1:Z7-1),VC(1:X-1),V(8)(1:Z8-1)
 ELSE  IF(L.EQ.9)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1),
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
*,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
  WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),V(7)(1:Z7-1)
*,VC(1:X-1),V(8)(1:Z8-1),VC(1:X-1),V(9)(1:Z9-1)
 ELSE  IF(L.EQ.10)THEN
  WRITE(11,'(6X,''IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
*''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
*,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
```

```
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(7)(1:Z7-1)
     *,VC(1:X-1),V(8)(1:Z8-1),VC(1:X-1),V(9)(1:Z9-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')
     *VC(1:X-1),V(10)(1:Z10-1)
      END IF
      IF(CB.EQ.1)GO TO 45
      WRITE(11,'(6X,A)')SUBSTR(VAS,Z,74-Z)
      IF(CB.GT.0)GO TO 45
      GO TO 100
C     CUANDO ES EL SEGUNDO IF (VA CON ELSE IF)
  120 IF(L.EQ.1)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,'')THEN'')')
     *VC(1:X-1),V(1)(1:Z1-1)
      ELSE IF(L.EQ.2)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,'')THEN'')')VC(1:X-1),V(1)(1:Z1-1),VC(1:X-1),V(2)(1:Z2-1)
      ELSE IF(L.EQ.3)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,  ''.OR.'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1 ),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      ELSE IF(L.EQ.4)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),
     *V(4)(1:Z4-1)
      ELSE IF(L.EQ.5)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,'')
     *THEN'')')VC(1:X-1),V(4)(1:Z4-1),VC(1:X-1),V(5)(1:Z5-1)
      ELSE IF(L.EQ.6)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')THEN'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      ELSE IF(L.EQ.7)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
```

```
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),
     *V(7)(1:Z7-1)
      ELSE IF(L.EQ.8)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,'')
     *THEN'')')VC(1:X-1),V(7)(1:Z7-1),VC(1:X-1),V(8)(1:Z8-1)
      ELSE IF(L.EQ.9)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,'')THEN'')')VC(1:X-1),V(7)(1:Z7-1)
     *,VC(1:X-1),V(8)(1:Z8-1),VC(1:X-1),V(9)(1:Z9-1)
      ELSE IF(L.EQ.10)THEN
      WRITE(11,'(6X,''ELSE IF('',A,''.EQ.'',A,''.OR.'',A,''.EQ.''
     *,A,''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(1)(1:Z1-1)
     *,VC(1:X-1),V(2)(1:Z2-1),VC(1:X-1),V(3)(1:Z3-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(4)(1:Z4-1)
     *,VC(1:X-1),V(5)(1:Z5-1),VC(1:X-1),V(6)(1:Z6-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,''.OR.'',A,''.EQ.'',A,
     *''.OR.'',A,''.EQ.'',A,''.OR.'')')VC(1:X-1),V(7)(1:Z7-1)
     *,VC(1:X-1),V(8)(1:Z8-1),VC(1:X-1),V(9)(1:Z9-1)
      WRITE(11,'(5X,''*'',A,''.EQ.'',A,'')THEN'')')
     *VC(1:X-1),V(10)(1:Z10-1)
      END IF
      IF(CB.EQ.1)GO TO 45
      WRITE(11,'(6X,A)')SUBSTR(VAS,Z,74-Z)
      IF(CB.GT.0)GO TO 45
      GO TO 100
105   STOP ' ERROR!!... FALTO "END CASE"'
106   STOP ' ERROR!!... FALTO CERRAR EL BEGIN ABIERTO "END BEGIN"'
150   WRITE(11,'(6X,''END IF'')')
      RETURN
      END
```

* **PARTE 3**: Programas para la declaración de variables.

[illegible]BCTES.VAR

```
      SUBROUTINE VAR(L1,L2)
C     SUBRUTINA PARA CONVERTIR LAS DECLARACIONES DE VARIABLES
C     Y ARREGLOS A SU CORRESPONDIENTE DIMENSION EN FORTRAN
      IMPLICIT INTEGER(A-Z)
      COMMON /VARBLE/VARS
      CHARACTER L1*6,L2*74,CADE*70,V*6(10),VV*6(2),VARS*6(100,5)
      CHARACTER ARR*1,TYPE*3
      ARR=' '
      TYPE='   '
   90 READ(10,'(A6,A74)',END=1100)L1,L2
      L1=UPPERC(L1)
      L2=UPPERC(L2)
C     PUEDEN EXISTIR LINEAS DE COMENTARIOS EN FTN
      DO 1 I=1,5
       IF(SUBSTR(L1,I,1).NE.' ')THEN
       DO 2 J=1,74
    2  IF(SUBSTR(L2,J,1).NE.' ')GO TO 3
       GO TO 90
    3  WRITE(11,'(''C'',A,A74)')SUBSTR(L1,I+1,6-I),SUBSTR(L2,J,
     *74-J)
       GO TO 90
       ENDIF
    1 CONTINUE
      DO 4 I=1,74
    4  IF(SUBSTR(L2,I,1).NE.' ')GO TO 6
C     LINEA EN BLANCO
      GO TO 90
    6 IF(SUBSTR(L2,I,4).EQ.'ENDA'.OR.SUBSTR(L2,I,4).EQ.'ENDV')
     *RETURN
      IF(SUBSTR(L2,I,5).EQ.'END A'.OR.SUBSTR(L2,I,5).EQ.'END V')
     *RETURN
C     SE OBTIENE UNA CADENA DE CARACTERES
      IC=I
      DO 8 I=IC,74
    8  IF(SUBSTR(L2,I,1).EQ.':')GO TO 10
      WRITE(6,'('' ERROR!!... FALTARON (:) EN DEC. VAR'')')
      STOP 'CHECAR "ENDV" O "ENDA" '
   10 CADE=SUBSTR(L2,IC,I-IC)
      DO 12 J=I,74
       IF(SUBSTR(L2,J,3).EQ.'MAT')THEN
          ARR='M'
          CALL MAT(L2,J,CADE,V,VV,II,TYPE)
          GO TO 89
        END IF
       IF(SUBSTR(L2,J,3).EQ.'VEC')THEN
         ARR='V'
         CALL VEC(L2,J,CADE,V,VV,II,TYPE)
         GO TO 89
       END IF
```

```
      IF(SUBSTR(L2,J,3).EQ.'NTE')THEN
         ARR='E'
         TYPE='INT'
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''INTEGER '',A)')CADE
         GO TO 89
      END IF
      IF(SUBSTR(L2,J,3).EQ.'REA')THEN
         ARR='E'
         TYPE='REA'
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''REAL '',A)')CADE
         GO TO 89
      END IF
      IF(SUBSTR(L2,J,3).EQ.'ARA'.OR.SUBSTR(L2,J,4).EQ.'CHAR')
     $ THEN
         ARR='C'
         TYPE='    '
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''CHARACTER '',A)')CADE
         GO TO 89
      END IF
      IF(SUBSTR(L2,J,3).EQ.'LOG'.OR.SUBSTR(L2,J,4).EQ.'BOOL')
     $ THEN
         ARR='L'
         TYPE='    '
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''LOGICAL '',A)')CADE
         GO TO 89
      END IF
      IF(SUBSTR(L2,J,4).EQ.'COMP')THEN
         ARR='X'
         TYPE='    '
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''COMPLEX '',A)')CADE
         GO TO 89
      END IF
      IF(SUBSTR(L2,J,4).EQ.'DOUB')THEN
         ARR='D'
         TYPE='    '
         CALL SEPARA(V,L2,II)
         WRITE(11,'(6X,''DOUBLE PRECISION '',A)')CADE
         GO TO 89
      END IF
   12 CONTINUE
      WRITE(6,'('' ERROR!!... "ARRAY" O "VAR" SIN TIPO'')')
      STOP '          CHECAR SINTAXIS '
   89 DO 21 J=1,100
   21 IF(VARS(J,1).EQ.'          ')GO TO 11
   11 JJ=J+II-1
      LL=0
```

```
      DO 5 J1=J,JJ
         LL=LL+1
         VARS(J1,1)=V(LL)
         VARS(J1,2)=ARR
         VARS(J1,3)=VV(1)
         VARS(J1,4)=VV(2)
    5    VARS(J1,5)=TYPE
      GO TO 90
1100  STOP
      END
```

[illegible]

```
      SUBROUTINE MAT(L2,J,CADE,V,VV,II,TYPE)
      CHARACTER L2*74,V*6(10),VV*6(2),CADE*70,DIM*70,TYPE*3
      IC=0
      DO 50 L=(J+3),74
      IF(SUBSTR(L2,L,1).EQ.'[' .OR. SUBSTR(L2,L,1).EQ.'(')THEN
        N=L+3
        IF(SUBSTR(L2,N,1).NE.',')THEN
          WRITE(6,'('' ERROR!!... EN DECLARACION DE MATRICES'')')
          STOP ' MAT[1<.>N,1..MJ ---> MAT[1..N,1..MJ'
        END IF
        IF(SUBSTR(L2,N+1,1).EQ.',')THEN
          WRITE(6,'('' ERROR!!... EN DECLARACION DE MATRICES PUNTO
     *DE MAS'')')
          STOP '          MAT[1<...>N,1..MJ ---> MAT[1..N,1..MJ'
        END IF
        N=N+1
        DO 3 N1=1,10
          IF(SUBSTR(L2,N+N1,1).EQ.',')THEN
            VV(1)=SUBSTR(L2,N,N1)
            N=N+N1+3
            IF(SUBSTR(L2,N,1).NE.',')THEN
              WRITE(6,'('' ERROR!!... EN DECLARACION DE
     *MATRICES'')')
              STOP ' MAT[1..N,1<.>MJ ---> MAT[1..N,1..MJ'
            END IF
            IF(SUBSTR(L2,N+1,1).EQ.',')THEN
              WRITE(6,'('' ERROR!!... EN DECLARACION DE MATRICES
     *PUNTO DE MAS'')')
              STOP '      MAT[1..N,1<...>MJ ---> MAT[1..N,1..MJ'
            END IF
            N=N+1
```

```
          DO 5 N2=1,10
             IF(SUBSTR(L2,N+N2,1).EQ.'3'.OR.SUBSTR(L2,N+N2,1)
  $.EQ.')')THEN
                VV(2)=SUBSTR(L2,N,N2)
                N=N+N2+1
                GO TO 10
             END IF
 5        CONTINUE
          STOP ' ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN
  $"MAT" '
          END IF
 3    CONTINUE
      STOP ' ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "MAT"'
    END IF
50  CONTINUE
    STOP 'ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "MAT"'
10  CALL SEPARA(V,L2,II)
    CALL DIMM(V,DIM,VV,II)
    DO 15 I=N,74
     IF(SUBSTR(L2,I,3).EQ.'NTE')THEN
       TYPE='INT'
       WRITE(11,'(6X,''INTEGER '',A,/,6X,''DIMENSION '',A)')
  $CADE,DIM
       RETURN
     END IF
     IF(SUBSTR(L2,I,3).EQ.'REA')THEN
       TYPE='REA'
       WRITE(11,'(6X,''REAL '',A,/,6X,''DIMENSION '',A)')
  $CADE,DIM
       RETURN
     END IF
    IF(SUBSTR(L2,I,3).EQ.'ARA'.OR.SUBSTR(L2,I,4).EQ.'CHAR')
  $THEN
       TYPE='CHA'
       WRITE(11,'(6X,''CHARACTER '',A)')DIM
       RETURN
     END IF
15  CONTINUE
    STOP ' ERROR!!... EN "MAT" TIPO DE VARIABLE NO CONSIDERADO'
    END
```

K[illegible]CTES.VEC

```
      SUBROUTINE VEC(L2,J,CADE,V,VV,II,TYPE)
      CHARACTER L2*74,CADE*70,V*6(10),VV*6(2),DIM*70,TYPE*3
      VV(2)='      '
      IC=0
      DO 50 L=(J+3),74
    1 IF(SUBSTR(L2,L,1).EQ.'[' .OR. SUBSTR(L2,L,1).EQ.'(')THEN
        N=L+3
        IF(SUBSTR(L2,N,1).NE.',')THEN
          WRITE(6,'('' ERROR!!... EN DECLARACION DE VECTORES'')')
          STOP '          VEC[1<.>N] --> VEC[1..N]'
        END IF
        IF(SUBSTR(L2,N+1,1).EQ.',')THEN
          WRITE(6,'('' ERROR!!... EN DECLARACION DE VECTORES PUNTO
     *DE MAS'')')
          STOP '            VEC[1<...>N] --> VEC[1..N]'
        END IF
        N=N+1
        DO 3 N1=1,10
        IF(SUBSTR(L2,N+N1,1).EQ.']'.OR.SUBSTR(L2,N+N1,1).EQ.')')
     * THEN
          VV(1)=SUBSTR(L2,N,N1)
          N=N+N1+1
          GO TO 10
        END IF
    3   CONTINUE
      STOP ' ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "VEC" '
      END IF
   50 CONTINUE
      STOP 'ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "VEC"'
   10 CALL SEPARA(V,L2,II)
      CALL DIMV(V,DIM,VV,II)
      DO 15 I=N,74
       IF(SUBSTR(L2,I,3).EQ.'NTE')THEN
         TYPE='INT'
         WRITE(11,'(6X,''INTEGER '',A,/,6X,''DIMENSION '',A)')
     *CADE,DIM
         RETURN
       END IF
       IF(SUBSTR(L2,I,3).EQ.'REA')THEN
         TYPE='REA'
         WRITE(11,'(6X,''REAL '',A,/,6X,''DIMENSION '',A)')
     *CADE,DIM
         RETURN
       END IF
```

```
      IF(SUBSTR(L2,I,3).EQ.'ARA'.OR.SUBSTR(L2,I,4).EQ.'CHAR')
     #THEN
        TYPE='CHA'
        WRITE(11,'(6X,''CHARACTER '',A)')DIM
        RETURN
      END IF
   15 CONTINUE
      STOP ' ERROR!!... EN "VEC" TIPO DE VARIABLE NO CONSIDERADO'
      END
```

[illegible].SEPARA

```
      SUBROUTINE SEPARA(V,L2,II)
      CHARACTER L2*74,V*6(10)
      J=1
      II=0
   10 DO 5 K=J,74
       IF(SUBSTR(L2,K,1).NE.' ')THEN
         DO 7 I=1,6
          IF(SUBSTR(L2,K+I,1).EQ.',')THEN
            II=II+1
            V(II)=SUBSTR(L2,K,I)
            J=K+I+1
            GO TO 10
          ELSE IF(SUBSTR(L2,K+I,1).EQ.':')THEN
C PUEDEN EXISTIR ESPACIOS ENTRE LA ULTIMA VAR Y LOS DOS PUNTOS
                DO 9 KK=1,6
    9            IF(SUBSTR(L2,K+I-KK,1).NE.' '.AND.SUBSTR(L2,
     $K+I-KK,1).NE.',')GO TO 12
                STOP ' ERROR!!... SINTAXIS EN DEC. DE "VAR" O
     $"ARRAY" '
   12           II=II+1
                V(II)=SUBSTR(L2,K,I-KK+1)
                J=K+I+1
                RETURN
          END IF
    7     CONTINUE
          WRITE(6,'('' ERROR!!... EN "VAR" O "ARRAY" VARIABLE MAYOR
     $DE 6 DIGITOS'')')
          STOP
       END IF
    5 CONTINUE
      STOP ' ERROR!!... EN DECLARACION DE "VAR" O "ARRAY"'
      END
```

KBBSBBCTES.DIMM

```
      SUBROUTINE DIMM(V,DIM,VV,II)
      CHARACTER V*6(10),DIM*70,VV*6(2),DIM1*70
      JL=INDEX(VV(1),' ')
      IF(JL.EQ.0)JL=7
      KL=INDEX(VV(2),' ')
      IF(KL.EQ.0)KL=7
      IL=INDEX(V(1),' ')
      IF(IL.EQ.0)IL=7
      DIM=V(1)(1:IL-1)//'('//VV(1)(1:JL-1)//','//VV(2)(1:KL-1)
     *//')'
      IF(II.EQ.1)RETURN
      DO 1 I=2,II
        DIM1=DIM
        LL=INDEX(DIM1,' ')
        IF(LL.EQ.0)LL=7
        IL=INDEX(V(I),' ')
        IF(IL.EQ.0)IL=7
    1 DIM=DIM1(1:LL-1)//','//V(I)(1:IL-1)//'('//VV(1)(1:JL-1)
     *//','//VV(2)(1:KL-1)//')'
      RETURN
      END
```

KBBSBBCTES.DIMV

```
      SUBROUTINE DIMV(V,DIM,VV,II)
      CHARACTER V*6(10),DIM*70,VV*6(2),DIM1*70
      JL=INDEX(VV(1),' ')
      IF(JL.EQ.0)JL=7
      KL=INDEX(VV(2),' ')
      IF(KL.EQ.0)KL=7
      IL=INDEX(V(1),' ')
      IF(IL.EQ.0)IL=7
      DIM=V(1)(1:IL-1)//'('//VV(1)(1:JL-1)//')'
      IF(II.EQ.1)RETURN
      DO 1 I=2,II
        DIM1=DIM
        LL=INDEX(DIM1,' ')
        IF(LL.EQ.0)LL=7
        IL=INDEX(V(I),' ')
        IF(IL.EQ.0)IL=7
    1 DIM=DIM1(1:LL-1)//','//V(I)(1:IL-1)//'('//VV(1)(1:JL-1)
     *//')'
      RETURN
      END
```

* **PARTE 4**: Programas para las primitivas de MATfor.

[illegible].PRIMI

```
      SUBROUTINE PRIMI(L2,L1,E)
      IMPLICIT INTEGER (A-Z)
      CHARACTER L2*74,VT*10,OPE*1,SIG*1,V*6(3),L1*6
      SIG=' '
      OPE=' '
      DO 1 Z=1,74
    1 IF(SUBSTR(L2,Z,1).NE.' ')GO TO 3
    3 DO 5 W=1,10
      IF(SUBSTR(L2,Z+W,1).EQ.'=')THEN
        VT=SUBSTR(L2,Z,W)
        L=INDEX(VT,' ')
        IF(L.GE.8)THEN
          WRITE(6,'('' ERROR!!... VARIABLE MAYOR DE 6 DIGITOS'')')
          STOP ' AL USAR FUNCION EXCLUSIVA DE *MATFOR* '
        END IF
        V(1)=VT(1:L-1)
        Z=Z+W+1
        GO TO 7
      ELSE IF(SUBSTR(L2,Z+W-1,1).EQ.'&'.OR.SUBSTR(L2,Z+W-1,1).EQ.
     *'?')THEN
C  PARTE PARA UNA FUNCION UNIVOCA
        SIG=SUBSTR(L2,Z+W-1,1)
        Z=Z+W-1
        DO 8 W1=1,10
    8   IF(SUBSTR(L2,Z+W1,1).NE.' ')GO TO 89
        STOP ' ERROR!!... FUNCION DE *MATFOR* DECLARADA INCOMPLE
     *TA'
   89   Z=Z+W1
        DO 10 W1=1,6
        IF(SUBSTR(L2,Z+W1,1).EQ.' ')THEN
          IF(SUBSTR(L2,Z+W1-1,1).EQ.'@')THEN
            V(1)=SUBSTR(L2,Z,W1-1)
            V(2)='@'
            V(3)=' '
          ELSE IF(SUBSTR(L2,Z+W1-1,1).NE.'@')THEN
            V(1)=SUBSTR(L2,Z,W1)
            V(2)=' '
            V(3)=' '
          END IF
          GO TO 12
        END IF
   10   CONTINUE
        STOP ' ERROR!!... FUNCION UNIVOCA DE *MATFOR* MAYOR DE 6
     *DIGITOS '
       END IF
    5  CONTINUE
       STOP ' ERROR!!... NO SE ENCONTRO SIGNO DE (=) EN FUNCION DE
     * *MATFOR* O > BLANCOS '
```

```
     7 DO 15 W=1,10
       IF(SUBSTR(L2,Z+W,1).EQ.'+'.OR.SUBSTR(L2,Z+W,1).EQ.'-'.OR.
      $SUBSTR(L2,Z+W,1).EQ.'*'.OR.SUBSTR(L2,Z+W,1).EQ.'/')THEN
          OPE=SUBSTR(L2,Z+W,1)
          VT=SUBSTR(L2,Z,W)
          DO 50 X=1,10
    50    IF(SUBSTR(VT,X,1).NE.' ')GO TO 52
          STOP ' ERROR!!... FUNCION DE $MATFOR$ DECLARADA INCOMPLETA
      $ CAMPO DE VARIABLE EN BLANCO'
    52    VT=SUBSTR(VT,X,11-X)
          L=INDEX(VT,' ')
          IF(L.GE.8)THEN
            WRITE(6,'('' ERROR!!... VARIABLE MAYOR DE 6 DIGITOS'')')
            STOP ' AL USAR FUNCION EXCLUSIVA DE $MATFOR$ '
          END IF
          V(2)=VT(1:L-1)
          Z=Z+W
          GO TO 9
        ELSE IF(SUBSTR(L2,Z+W-1,1).EQ.'>'.OR.SUBSTR(L2,Z+W-1,1)
       $.EQ.'<'.OR.SUBSTR(L2,Z+W-1,1).EQ.'#'.OR.SUBSTR(L2,
       $Z+W-1,1).EQ.'$')THEN
          V(2)=SUBSTR(L2,Z+W-1,1)
          Z=Z+W-1
          DO 20 W1=1,10
    20    IF(SUBSTR(L2,Z+W1,1).NE.' ')GO TO 22
          STOP ' ERROR!!... FUNCION DE $MATFOR$ DECLARADA
      $INCOMPLETA'
    22    Z=Z+W1
          DO 25 W1=1,6
    25    IF(SUBSTR(L2,Z+W1,1).EQ.' ')GO TO 24
          WRITE(6,'('' ERROR!!... VARIABLE MAYOR DE 6 DIGITOS '')')
          STOP ' AL USAR FUNCION EXCLUSIVA DE $MATFOR$ '
    24    V(3)=SUBSTR(L2,Z,W1)
          Z=Z+W1
          GO TO 12
        END IF
    15  CONTINUE
        STOP ' ERROR!!... FUNCION DE $MATFOR$ DECLARADA INCOMPLETA'
     9  DO 30 W1=1,10
    30  IF(SUBSTR(L2,Z+W1,1).NE.' ')GO TO 32
        STOP ' ERROR!!... FUNCION DE $MATFOR$ DECLARADA INCOMPLETA'
    32  Z=Z+W1
        DO 35 W1=1,6
    35  IF(SUBSTR(L2,Z+W1,1).EQ.' ')GO TO 34
        WRITE(6,'('' ERROR!!... VARIABLE MAYOR DE 6 DIGITOS '')')
        STOP ' AL USAR FUNCION EXCLUSIVA DE $MATFOR$ '
    34  V(3)=SUBSTR(L2,Z,W1)
C   LLAMADA PARA INVESTIGAR SI SE TRATA DE VECTORES O MATRICES
    12  CALL QUESOY(V,SIG,OPE,L1,E)
C   ********************************
        RETURN
        END
```

K[illegible]CTES.QUESOY

```
      SUBROUTINE QUESOY(V,SIG,OPE,L1,E)
C SE VA FORMAR LA FUNCIONE p.e. V=V+V   O  M=M*M
      IMPLICIT INTEGER (A-Z)
      COMMON /VARBLE/VARS
      CHARACTER*1 VARS*6(100,5),V*6(3),DI*6(3,2),FUN*5,F1,F2,F3,
     *SIG,OPE
      CHARACTER*6 TIPO,TIPO1,TIPO2,TIPO3,L1*6
      BAN=0
      IF(SIG.EQ.' ')GO TO 10
C FUNCION UNIVOCA
      DO 3 Y=1,100
      IF(V(1).EQ.VARS(Y,1))THEN
        FUN=SIG//VARS(Y,2)(1:1)//V(2)
        DI(1,1)=VARS(Y,3)
        DI(1,2)=VARS(Y,4)
        TIPO=VARS(Y,5)
        GO TO 60
      END IF
    3 CONTINUE
      WRITE(6,'(' ERROR!!... VARIABLE NO DECLARADA EN "VAR" O
     *"ARRAY"')')
      STOP ' Y UTILIZADA EN FUNCIONES DE *MATFOR*'
   10 DO 5 Y=1,100
    5 IF(V(1).EQ.VARS(Y,1)) GO TO 15
      WRITE(6,'(' ERROR!!... VARIABLE NO DECLARADA EN "VAR" O
     *"ARRAY"')')
      STOP ' Y UTILIZADA EN FUNCIONES DE *MATFOR*'
   15 TIPO1=VARS(Y,5)
      F1=VARS(Y,2)
      DI(1,1)=VARS(Y,3)
      DI(1,2)=VARS(Y,4)
      IF(V(2).EQ.'>'.OR.V(2).EQ.'<'.OR.V(2).EQ.'#'.OR.V(2).EQ.'$')
     *THEN
        F2=V(2)
        BAN=1
        GO TO 8
      END IF
      DO 6 Y=1,100
    6 IF(V(2).EQ.VARS(Y,1)) GO TO 16
      WRITE(6,'(' ERROR!!... VARIABLE NO DECLARADA EN "VAR" O
     *"ARRAY"')')
      STOP ' Y UTILIZADA EN FUNCIONES DE *MATFOR*'
   16 TIPO2=VARS(Y,5)
      F2=VARS(Y,2)
      DI(2,1)=VARS(Y,3)
      DI(2,2)=VARS(Y,4)
    8 DO 7 Y=1,100
    7 IF(V(3).EQ.VARS(Y,1)) GO TO 17
      WRITE(6,'(' ERROR!!... VARIABLE NO DECLARADA EN "VAR" O
     *"ARRAY"')')
      STOP ' Y UTILIZADA EN FUNCIONES DE *MATFOR*'
```

```
   17 TIPO3=VARS(Y,5)
      IF(BAN.EQ.0)THEN
        IF(TIPO1.EQ.TIPO2.AND.TIPO3.EQ.TIPO1)GO TO 59
        WRITE(6,'('' ERROR!!... VARIABLES DE DIFERENTE TIPO EN
     $FUNCIONES DE $MATFOR'')')
        STOP ' REALES/ENTERAS??? '
      ELSE IF(BAN.NE.0)THEN
        IF(TIPO1.EQ.TIPO3)GO TO 59
        WRITE(6,'('' ERROR!!... VARIABLES DE DIFERENTE TIPO EN
     $FUNCIONES DE $MATFOR'')')
        STOP ' REALES/ENTERAS??? '
      END IF
   59 TIPO=TIPO1
      F3=VARS(Y,2)
      DI(3,1)=VARS(Y,3)
      DI(3,2)=VARS(Y,4)
      FUN=F1//'='//F2//F3
      IF(OPE.NE.' ')FUN=F1//'='//F2//OPE//F3
   60 CALL FUNS(FUN,V,DI,TIPO,L1,E)
C ************************************
      RETURN
      END
```

K***CTES.FUNS**

```
      SUBROUTINE FUNS(FUN,V,DI,TIPO,L1,E)
      IMPLICIT INTEGER (A-Z)
      CHARACTER FUN*5,V*6(3),FUNCIO*6(22,2),DI*6(3,2),FN*6,TIPO*6
      CHARACTER L1*6,ENUM*6
      DATA (FUNCIO(I,1),I=1,11)/'V=V+V','V=V-V','V=V*V','V=V*M',
     *'V=V*E','V=E*V','E=>V','E=<V','E=#V','M=M+M','M=M-M'/
      DATA (FUNCIO(I,1),I=12,22)/'M=M*M','V=M*V','M=M*E','M=E*M',
     *'M=*M','&V','&V@','?V','?V@','&M','?M'/
      DATA (FUNCIO(I,2),I=1,11)/'SUMVEC','RESVEC','MULVEC',
     *'MVMVEC','MVEVEC','MEVVEC','MAXVEC','MINVEC','MEDVEC',
     *'SUMMAT','RESMAT'/
      DATA (FUNCIO(I,2),I=12,22)/'MULMAT','MMVMAT','MMEMAT',
     *'MEMMAT','INVMAT','IMPVEC','IMVVEC','LEEVEC','LEVVEC',
     *'IMPMAT','LEEMAT'/
      ENUM=SUBSTR(L1,E+1,6-E)
      DO 2 Z=1,22
      IF(FUN.EQ.FUNCIO(Z,1))THEN
        CALL COMPAT(FUN,DI)
        CALL SUBSTA(FUNCIO(Z,2),NST)
        IF(NST.EQ.0)CALL BIBLIO(FUNCIO(Z,2),Z,TIPO)
        GO TO 10
      END IF
    2 CONTINUE
      WRITE(6,'('' ERROR!!... FUNCION NO DISPONIBLE *'',A,''*''
     *)')FUN
      STOP
```

```
10 FN=FUNCID(Z,1)
   L=INDEX(V(1),' ')
   M=INDEX(V(2),' ')
   N=INDEX(V(3),' ')
   IF(L.EQ.0)L=7
   IF(M.EQ.0)M=7
   IF(N.EQ.0)N=7
   IF(FN.EQ.'V=V+V'.OR.FN.EQ.'V=V-V'.OR.FN.EQ.'V=V$V'
  $.OR.FN.EQ.'V=V$E'.OR.FN.EQ.'V=E$V'.OR.FN.EQ.'E=>V'.OR.
  $FN.EQ.'E=<V '.OR.FN.EQ.'E=#V')THEN
       IF(FUN.EQ.'V=V$E')THEN
       WRITE(11,'(A6,''ID$1='',A)')ENUM,DI(1,1)
       GO TO 99
       END IF
       WRITE(11,'(A6,''ID$1='',A)')ENUM,DI(3,1)
       IF(V(2).EQ.'>'.OR.V(2).EQ.'<'.OR.V(2).EQ.'#')THEN
         WRITE(11,'(6X,''CALL '',A,'' ('',A,'','',A,'',ID$1)'')')
  $FUNCID(Z,2),V(1)(1:L-1),V(3)(1:N-1)
       RETURN
       END IF
99     WRITE(11,'(6X,''CALL '',A,'' ('',A,'','',A,'','',A,'',
  $ID$1)'')')FUNCID(Z,2),V(1)(1:L-1),V(2)(1:M-1),V(3)(1:N-1)
       RETURN
   END IF
   IF(FN.EQ.'&V'.OR.FN.EQ.'&V@'.OR.FN.EQ.'?V'.OR.FN.EQ.'?V@')
  $THEN
       WRITE(11,'(A6,''ID$1='',A)')ENUM,DI(1,1)
       WRITE(11,'(6X,''CALL '',A,'' ('',A,'',ID$1)'')')
  $FUNCID(Z,2),V(1)(1:L-1)
       RETURN
   END IF
   IF(FN.EQ.'M=M+M'.OR.FN.EQ.'M=M-M')THEN
       WRITE(11,'(A6,''ID$1='',A,/,6X,''ID$2='',A)')
  $ENUM,DI(1,1),DI(1,2)
       WRITE(11,'(6X,''CALL '',A,'' ('',A,'','',A,'','',A,'',
  $ID$1,ID$2)'')')FUNCID(Z,2),V(1)(1:L-1),V(2)(1:M-1),
  $V(3)(1:N-1)
       RETURN
   END IF
   IF(FN.EQ.'M=M$M')THEN
       WRITE(11,'(A6,''ID$1='',A,/,6X,''ID$2='',A,/,6X,
  $''ID$3='',A)')ENUM,DI(1,1),DI(3,2),DI(2,2)
       WRITE(11,'(6X,''CALL '',A,'' ('',A,'','',A,'','',A,'',
  $ID$1,ID$2,ID$3)'')')FUNCID(Z,2),V(1)(1:L-1),V(2)(1:M-1),
  $V(3)(1:N-1)
       RETURN
   END IF
```

```
 IF(FUN.EQ.'M=M*E'.OR.FUN.EQ.'M=E*M')THEN
     WRITE(11,'(A6,''ID$1='',A)')ENUM,DI(1,1)
     WRITE(11,'(6X,''ID$2='',A)')DI(1,2)
     WRITE(11,'(6X,''CALL '',A,''('',A,'','',A,'','',A,'',
*ID$1,ID$2)'')')FUNCIO(Z,2),V(1)(1:L-1),V(2)(1:M-1),
$V(3)(1:N-1)
     RETURN
 END IF
 IF(FN.EQ.'M=$M')THEN
     WRITE(11,'(A6,''ID$1='',A)')ENUM,DI(1,1)
     WRITE(11,'(6X,''CALL '',A,''('',A,'','',A,'',ID$1)'')')
$FUNCIO(Z,2),V(1)(1:L-1),V(3)(1:N-1)
     RETURN
 END IF
 IF(FN.EQ.'&M'.OR.FN.EQ.'?M')THEN
     WRITE(11,'(A6,''ID$1='',A,/,6X,''ID$2='',A)')
$ENUM,DI(1,1),DI(1,2)
     WRITE(11,'(6X,''CALL '',A,''('',A,'',ID$1,ID$2)'')')
$FUNCIO(Z,2),V(1)(1:L-1)
     RETURN
 END IF
 IF(FN.EQ.'V=V*M')THEN
     WRITE(11,'(A6,''ID$1='',A,/,6X,''ID$2='',A)')
$ENUM,DI(3,2),DI(3,1)
     WRITE(11,'(6X,''CALL '',A,''('',A,'','',A,'','',A,'',
*ID$1,ID$2)'')')FUNCIO(Z,2),V(1)(1:L-1),V(2)(1:M-1),
$V(3)(1:N-1)
     RETURN
 END IF
 IF(FN.EQ.'V=M*V')THEN
     WRITE(11,'(A6,''ID$1='',A,/,6X,''ID$2='',A)')
$ENUM,DI(2,1),DI(2,2)
     WRITE(11,'(6X,''CALL '',A,''('',A,'','',A,'','',A,'',
*ID$1,ID$2)'')')FUNCIO(Z,2),V(1)(1:L-1),V(2)(1:M-1),
$V(3)(1:N-1)
     RETURN
 END IF
 END
```

KBB*BBCTER.COMPAT

```
 SUBROUTINE COMPAT(FUN,DI)
 IMPLICIT INTEGER (A-Z)
 CHARACTER FUN*5,DI*6(3,2)
 IF(FUN.EQ.'V=V+V'.OR.FUN.EQ.'V=V-V'.OR.FUN.EQ.'V=V*V')THEN
 IF(DI(1,1).EQ.DI(2,1).AND.DI(2,1).EQ.DI(3,1))RETURN
 STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN
$ V=V+V,V=V-V,V=V*V'
 END IF
 IF(FUN.EQ.'V=V*M')THEN
    IF(DI(1,1).EQ.DI(3,2).AND.DI(2,1).EQ.DI(3,1))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V*M" '
 END IF
 IF(FUN.EQ.'V=V*E')THEN
    IF(DI(1,1).EQ.DI(2,1))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V*E" '
 END IF
 IF(FUN.EQ.'V=E*V')THEN
    IF(DI(1,1).EQ.DI(3,1))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=E*V" '
 END IF
 IF(FUN.EQ.'M=M+M'.OR.FUN.EQ.'M=M-M')THEN
    IF(DI(1,1).EQ.DI(2,1).AND.DI(1,1).EQ.DI(3,1).AND.
$DI(1,2).EQ.DI(2,2).AND.DI(1,2).EQ.DI(3,2))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M+M O
$M=M-M"'
 END IF
 IF(FUN.EQ.'M=M*M')THEN
    IF(DI(1,1).EQ.DI(2,1).AND.DI(2,2).EQ.DI(3,1).AND.
$DI(1,2).EQ.DI(3,2))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M*M" '
 END IF
 IF(FUN.EQ.'V=M*V')THEN
    IF(DI(1,1).EQ.DI(2,1).AND.DI(2,2).EQ.DI(3,1))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=M*V" '
 END IF
 IF(FUN.EQ.'M=M*E')THEN
    IF(DI(1,1).EQ.DI(2,1).AND.DI(1,2).EQ.DI(2,2))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M*E" '
 END IF
 IF(FUN.EQ.'M=E*M')THEN
    IF(DI(1,1).EQ.DI(3,1).AND.DI(1,2).EQ.DI(3,2))RETURN
    STOP ' ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=E*M" '
 END IF
 IF(FUN.EQ.'M=*M')THEN
    IF(DI(1,1).EQ.DI(1,2).AND.DI(3,1).EQ.DI(3,2).AND.
*DI(1,1).EQ.DI(3,1))RETURN
    WRITE(6,'('' ERROR!!... DIMENSIONES INCOMPATIBLES '')')
    STOP ' MATRIZ CUADRADA PARA OBTENER SU INVERSA'
 END IF
```

```
      IF(FUN.EQ.'E=>V'.OR.FUN.EQ.'E=<V'.OR.FUN.EQ.'E=#V'.OR.
     #FUN.EQ.'&V'.OR.FUN.EQ.'&V@'.OR.FUN.EQ.'?V'.OR.FUN.EQ.
     *'?V@'.OR.FUN.EQ.'&M'.OR.FUN.EQ.'?M')RETURN
      STOP ' ERROR!!... FUNCION NO CONSIDERADA EN SUB-COMPAT'
      END
```

KB0:SUBCTES.SUBSTA

```
      SUBROUTINE SUBSTA(FUNC,NST)
      COMMON /STATUS/STA
      CHARACTER STA*6(22,2),FUNC*6
C SE VA A INVESTIGAR SI LA FUNCION ES PEDIDA POR PRIMERA VES O NO
      DO 1 I=1,22
      IF(FUNC.EQ.STA(I,1))THEN
         NST=0
         IF(STA(I,2).EQ.'1')NST=1
         RETURN
      END IF
    1 CONTINUE
      WRITE(6,'(' ERROR!!... FUNCION NO CONSIDERADA '',A,'')FUNC
      STOP
      END
```

KB0:SUBCTES.BIBLIO

```
      SUBROUTINE BIBLIO(FUN,IN,TIPO)
      COMMON /STATUS/STA
      CHARACTER FUN*6,STA*6(22,2),TIPO*6
      IF(FUN.EQ.'SUMVEC')THEN
       IF(TIPO.EQ.'INT')THEN
       WRITE(12,'(6X,''SUBROUTINE SUMVEC(IP1,IP2,IP3,IDIM)'')')
       WRITE(12,'(6X,''DIMENSION IP1(IDIM),IP2(IDIM),
     *IP3(IDIM)'')')
       WRITE(12,'(6X,''DO 70000 I=1,IDIM'')')
       WRITE(12,'(''70000 IP1(I)=IP2(I)+IP3(I)'')')
       WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
       STA(IN,2)='1'
       RETURN
       END IF
       WRITE(12,'(6X,''SUBROUTINE SUMVEC(RP1,RP2,RP3,IDIM)'')')
       WRITE(12,'(6X,''DIMENSION RP1(IDIM),RP2(IDIM),
     *RP3(IDIM)'')')
       WRITE(12,'(6X,''DO 70000 I=1,IDIM'')')
       WRITE(12,'(''70000 RP1(I)=RP2(I)+RP3(I)'')')
       WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
       STA(IN,2)='1'
       RETURN
      END IF
```

```
 IF(FUN.EQ.'RESVEC')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE RESVEC(IP1,IP2,IP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM),IP2(IDIM),
*IP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70001 I=1,IDIM'')')
  WRITE(12,'(''70001 IP1(I)=IP2(I)-IP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE RESVEC(RP1,RP2,RP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM),RP2(IDIM),
*RP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70001 I=1,IDIM'')')
  WRITE(12,'(''70001 RP1(I)=RP2(I)-RP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
 IF(FUN.EQ.'MULVEC')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MULVEC(IP1,IP2,IP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM),IP2(IDIM),
*IP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70002 I=1,IDIM'')')
  WRITE(12,'(''70002 IP1(I)=IP2(I)*IP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE MULVEC(RP1,RP2,RP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM),RP2(IDIM),
*RP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70002 I=1,IDIM'')')
  WRITE(12,'(''70002 RP1(I)=RP2(I)*RP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
 IF(FUN.EQ.'MVMVEC')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MVMVEC(IP1,IP2,IP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM1),IP2(IDIM1),
*IP3(IDIM1,IDIM2)'')')
  WRITE(12,'(6X,''DO 70003 I=1,IDIM1'')')
  WRITE(12,'(6X,''IP1(I)=0'')')
  WRITE(12,'(6X,''DO 70003 J=1,IDIM2'')')
  WRITE(12,'(''70003 IP1(I)=IP2(J)*IP3(J,I)+IP1(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
```

```
  WRITE(12,'(6X,''SUBROUTINE MVMVEC(RP1,RP2,RP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM1),RP2(IDIM1),RP3(IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DO 70003 I=1,IDIM1'')')
  WRITE(12,'(6X,''RP1(I)=0'')')
  WRITE(12,'(6X,''DO 70003 J=1,IDIM2'')')
  WRITE(12,'(''70003 RP1(I)=RP2(J)*RP3(J,I)+RP1(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
 IF(FUN.EQ.'MVEVEC')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MVEVEC(IP1,IP2,IP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM),IP2(IDIM)'')')
  WRITE(12,'(6X,''DO 70004 I=1,IDIM'')')
  WRITE(12,'(''70004 IP1(I)=IP2(I)*IP3'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE MVEVEC(RP1,RP2,RP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM),RP2(IDIM)'')')
  WRITE(12,'(6X,''DO 70004 I=1,IDIM'')')
  WRITE(12,'(''70004 RP1(I)=RP2(I)*RP3'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
 IF(FUN.EQ.'MEVVEC')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MEVVEC(IP1,IP2,IP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM),IP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70005 I=1,IDIM'')')
  WRITE(12,'(''70005 IP1(I)=IP2*IP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE MEVVEC(RP1,RP2,RP3,IDIM)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM),RP3(IDIM)'')')
  WRITE(12,'(6X,''DO 70005 I=1,IDIM'')')
  WRITE(12,'(''70005 RP1(I)=RP2*RP3(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
```

```
IF(FUN.EQ.'MAXVEC')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE MAXVEC(IP1,IP2,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION IP2(IDIM)'')')
 WRITE(12,'(6X,''IP1=IP2(1)'')')
 WRITE(12,'(6X,''DO 70006 I=2,IDIM'')')
 WRITE(12,'(''70006 IF(IP1.LT.IP2(I))IP1=IP2(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
 WRITE(12,'(6X,''SUBROUTINE MAXVEC(RP1,RP2,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION RP2(IDIM)'')')
 WRITE(12,'(6X,''RP1=RP2(1)'')')
 WRITE(12,'(6X,''DO 70006 I=2,IDIM'')');
 WRITE(12,'(''70006 IF(RP1.LT.RP2(I))RP1=RP2(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
IF(FUN.EQ.'MINVEC')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE MINVEC(IP1,IP2,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION IP2(IDIM)'')')
 WRITE(12,'(6X,''IP1=IP2(1)'')')
 WRITE(12,'(6X,''DO 70007 I=2,IDIM'')')
 WRITE(12,'(''70007 IF(IP1.GT.IP2(I))IP1=IP2(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
 WRITE(12,'(6X,''SUBROUTINE MINVEC(RP1,RP2,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION RP2(IDIM)'')')
 WRITE(12,'(6X,''RP1=RP2(1)'')')
 WRITE(12,'(6X,''DO 70007 I=2,IDIM'')')
 WRITE(12,'(''70007 IF(RP1.GT.RP2(I))RP1=RP2(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
IF(FUN.EQ.'MEDVEC')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE MEDVEC(IP1,IP2,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION IP2(IDIM)'')')
 WRITE(12,'(6X,''IP1=0'')')
 WRITE(12,'(6X,''DO 70008 I=1,IDIM'')')
 WRITE(12,'(''70008 IP1=IP2(I)+IP1'')')
 WRITE(12,'(6X,''IP1=IP1/IDIM'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
```

```
   WRITE(12,'(6X,''SUBROUTINE MEDVEC(RP1,RP2,IDIM)'')')
   WRITE(12,'(6X,''DIMENSION RP2(IDIM)'')')
   WRITE(12,'(6X,''RP1=0'')')
   WRITE(12,'(6X,''DO 70008 I=1,IDIM'')')
   WRITE(12,'(''70008 RP1=RP2(I)+RP1'')')
   WRITE(12,'(6X,''RP1=RP1/IDIM'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
  END IF
  IF(FUN.EQ.'SUMMAT')THEN
   IF(TIPO.EQ.'INT')THEN
   WRITE(12,'(6X,''SUBROUTINE SUMMAT(IP1,IP2,IP3,IDIM1,
*IDIM2)'')')
   WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2),IP2(IDIM1,
*IDIM2),IP3(IDIM1,IDIM2)'')')
   WRITE(12,'(6X,''DO 70009 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70009 J=1,IDIM2'')')
   WRITE(12,'(''70009 IP1(I,J)=IP2(I,J)+IP3(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
   END IF
   WRITE(12,'(6X,''SUBROUTINE SUMMAT(RP1,RP2,RP3,IDIM1,
*IDIM2)'')')
   WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2),RP2(IDIM1,
*IDIM2),RP3(IDIM1,IDIM2)'')')
   WRITE(12,'(6X,''DO 70009 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70009 J=1,IDIM2'')')
   WRITE(12,'(''70009 RP1(I,J)=RP2(I,J)+RP3(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
  END IF
  IF(FUN.EQ.'RESMAT')THEN
   IF(TIPO.EQ.'INT')THEN
   WRITE(12,'(6X,''SUBROUTINE RESMAT(IP1,IP2,IP3,IDIM1,
*IDIM2)'')')
   WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2),IP2(IDIM1,
*IDIM2),IP3(IDIM1,IDIM2)'')')
   WRITE(12,'(6X,''DO 70010 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70010 J=1,IDIM2'')')
   WRITE(12,'(''70010 IP1(I,J)=IP2(I,J)-IP3(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
   END IF
```

```
   WRITE(12,'(6X,''SUBROUTINE RESMAT(RP1,RP2,RP3,IDIM1,
*IDIM2)'')')
   WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2),RP2(IDIM1,
*IDIM2),RP3(IDIM1,IDIM2)'')')
   WRITE(12,'(6X,''DO 70010 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70010 J=1,IDIM2'')')
   WRITE(12,'(''70010 RP1(I,J)=RP2(I,J)-RP3(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
  END IF
  IF(FUN.EQ.'MULMAT')THEN
   IF(TIPO.EQ.'INT')THEN
   WRITE(12,'(6X,''SUBROUTINE MULMAT(IP1,IP2,IP3,IDIM1,IDIM2,
*IDIM3)'')')
   WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2),IP2(IDIM1,
*IDIM3),IP3(IDIM3,IDIM2)'')')
   WRITE(12,'(6X,''DO 70011 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70011 J=1,IDIM2'')')
   WRITE(12,'(6X,''IP1(I,J)=0'')')
   WRITE(12,'(6X,''DO 70011 K=1,IDIM3'')')
   WRITE(12,'(''70011 IP1(I,J)=IP2(I,K)*IP3(K,J)+
*IP1(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
   END IF
   WRITE(12,'(6X,''SUBROUTINE MULMAT(RP1,RP2,RP3,IDIM1,IDIM2,
*IDIM3)'')')
   WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2),RP2(IDIM1,
*IDIM3),RP3(IDIM3,IDIM2)'')')
   WRITE(12,'(6X,''DO 70011 I=1,IDIM1'')')
   WRITE(12,'(6X,''DO 70011 J=1,IDIM2'')')
   WRITE(12,'(6X,''RP1(I,J)=0'')')
   WRITE(12,'(6X,''DO 70011 K=1,IDIM3'')')
   WRITE(12,'(''70011 RP1(I,J)=RP2(I,K)*RP3(K,J)+
*RP1(I,J)'')')
   WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
   STA(IN,2)='1'
   RETURN
  END IF
```

```
 IF(FUN.EQ.'MMVMAT')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MMVMAT(IP1,IP2,IP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM1),IP2(IDIM1,IDIM2),
*IP3(IDIM2)'')')
  WRITE(12,'(6X,''DO 70012 I=1,IDIM1'')')
  WRITE(12,'(6X,''IP1(I)=0'')')
  WRITE(12,'(6X,''DO 70012 J=1,IDIM2'')')
  WRITE(12,'(''70012 IP1(I)=IP2(I,J)*IP3(J)+IP1(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE MMVMAT(RP1,RP2,RP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM1),RP2(IDIM1,IDIM2),
*RP3(IDIM2)'')')
  WRITE(12,'(6X,''DO 70012 I=1,IDIM1'')')
  WRITE(12,'(6X,''RP1(I)=0'')')
  WRITE(12,'(6X,''DO 70012 J=1,IDIM2'')')
  WRITE(12,'(''70012 RP1(I)=RP2(I,J)*RP3(J)+RP1(I)'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
 IF(FUN.EQ.'MMEMAT')THEN
  IF(TIPO.EQ.'INT')THEN
  WRITE(12,'(6X,''SUBROUTINE MMEMAT(IP1,IP2,IP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2),IP2(IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DO 70013 I=1,IDIM1'')')
  WRITE(12,'(6X,''DO 70013 J=1,IDIM2'')')
  WRITE(12,'(''70013 IP1(I,J)=IP2(I,J)*IP3'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
  END IF
  WRITE(12,'(6X,''SUBROUTINE MMEMAT(RP1,RP2,RP3,IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2),RP2(IDIM1,
*IDIM2)'')')
  WRITE(12,'(6X,''DO 70013 I=1,IDIM1'')')
  WRITE(12,'(6X,''DO 70013 J=1,IDIM2'')')
  WRITE(12,'(''70013 RP1(I,J)=RP2(I,J)*RP3'')')
  WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
  STA(IN,2)='1'
  RETURN
 END IF
```

```
      IF(FUN.EQ.'MEMMAT')THEN
       IF(TIPO.EQ.'INT')THEN
        WRITE(12,'(6X,''SUBROUTINE MEMMAT(IP1,IP2,IP3,IDIM1,
     *IDIM2)'')')
        WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2),IP3(IDIM1,
     *IDIM2)'')')
        WRITE(12,'(6X,''DO 70014 I=1,IDIM1'')')
        WRITE(12,'(6X,''DO 70014 J=1,IDIM2'')')
        WRITE(12,'(''70014 IP1(I,J)=IP2*IP3(I,J)'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
        END IF
        WRITE(12,'(6X,''SUBROUTINE MEMMAT(RP1,RP2,RP3,IDIM1,
     *IDIM2)'')')
        WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2),RP3(IDIM1,
     *IDIM2)'')')
        WRITE(12,'(6X,''DO 70014 I=1,IDIM1'')')
        WRITE(12,'(6X,''DO 70014 J=1,IDIM2'')')
        WRITE(12,'(''70014 RP1(I,J)=RP2*RP3(I,J)'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
       END IF
       IF(FUN.EQ.'INVMAT')THEN
        WRITE(12,'(6X,''SUBROUTINE INVMAT(RP1,RP2,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION RP1(IDIM,IDIM),RP2(IDIM,
     *IDIM)'')')
        WRITE(12,'(6X,''DIMENSION A(20,40),B(20,40),C(20,40),
     *D(20,40)'')')
        WRITE(12,'(6X,''DO 70015 I=1,IDIM'')')
        WRITE(12,'(6X,''DO 70015 J=1,IDIM*2'')')
        WRITE(12,'(''70015  A(I,J)=0 '')')
        WRITE(12,'(6X,''DO 70016 I=1,IDIM'')')
        WRITE(12,'(6X,''DO 70016 J=1,IDIM'')')
        WRITE(12,'(''70016  A(I,J)=RP2(I,J)'')')
C   SE AGREGA LA MATRIZ IDENTIDAD A LA MATRIZ A INVERTIR
        WRITE(12,'(6X,''DO 70017 I=1,IDIM'')')
        WRITE(12,'(''70017  A(I,IDIM+I)=1'')')
        WRITE(12,'(6X,''DO 70018 K=1,IDIM'')')
        WRITE(12,'(6X,''DO 70019 I=1,IDIM'')')
        WRITE(12,'(6X,''IF(K.NE.I)THEN'')')
        WRITE(12,'(6X,''C(K,K)=A(K,K)'')')
        WRITE(12,'(6X,''D(I,K)=A(I,K)'')')
        WRITE(12,'(6X,''DO 70020 J=1,IDIM*2'')')
        WRITE(12,'(6X,''A(K,J)=A(K,J)/C(K,K)'')')
        WRITE(12,'(6X,''B(I,J)=A(K,J)*D(I,K)'')')
        WRITE(12,'(''70020 A(I,J)=A(I,J)-B(I,J)'')')
        WRITE(12,'(6X,''END IF'')')
        WRITE(12,'(''70019 CONTINUE'')')
        WRITE(12,'(''70018 CONTINUE'')')
```

```
C SE PONE LA MATRIZ INVERSA EN EL PRIMER PARAMETRO
        WRITE(12,'(6X,''DO 70021 I=1,IDIM'')')
        WRITE(12,'(6X,''DO 70021 J=1,IDIM'')')
        WRITE(12,'(''70021 RP1(I,J)=A(I,J+IDIM)'')')
        WRITE(12,'(6X,''RETURN'')')
        WRITE(12,'(6X,''END'')')
        STA(IN,2)='1'
        RETURN
       END IF
       IF(FUN.EQ.'IMPVEC')THEN
        IF(TIPO.EQ.'INT')THEN
        WRITE(12,'(6X,''SUBROUTINE IMPVEC(IP1,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION IP1(IDIM)'')')
        WRITE(12,'(6X,''WRITE(6,*)IP1'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
        END IF
        WRITE(12,'(6X,''SUBROUTINE IMPVEC(RP1,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION RP1(IDIM)'')')
        WRITE(12,'(6X,''WRITE(6,*)RP1'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
       END IF
       IF(FUN.EQ.'IMVVEC')THEN
        IF(TIPO.EQ.'INT')THEN
        WRITE(12,'(6X,''SUBROUTINE IMVVEC(IP1,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION IP1(IDIM)'')')
        WRITE(12,'(6X,''DO 70022 I=1,IDIM'')')
        WRITE(12,'(''70022 WRITE(6,*)IP1(I)'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
        END IF
        WRITE(12,'(6X,''SUBROUTINE IMVVEC(RP1,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION RP1(IDIM)'')')
        WRITE(12,'(6X,''DO 70022 I=1,IDIM'')')
        WRITE(12,'(''70022 WRITE(6,*)RP1(I)'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
       END IF
       IF(FUN.EQ.'LEEVEC')THEN
        IF(TIPO.EQ.'INT')THEN
        WRITE(12,'(6X,''SUBROUTINE LEEVEC(IP1,IDIM)'')')
        WRITE(12,'(6X,''DIMENSION IP1(IDIM)'')')
        WRITE(12,'(6X,''READ(5,*)IP1'')')
        WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
        STA(IN,2)='1'
        RETURN
        END IF
```

```
 WRITE(12,'(6X,''SUBROUTINE LEEVEC(RP1,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION RP1(IDIM)'')')
 WRITE(12,'(6X,''READ(5,*)RP1'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
IF(FUN.EQ.'LEVVEC')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE LEVVEC(IP1,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION IP1(IDIM)'')')
 WRITE(12,'(6X,''DO 70023 I=1,IDIM'')')
 WRITE(12,'(''70023 READ(5,*)IP1(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
 WRITE(12,'(6X,''SUBROUTINE LEVVEC(RP1,IDIM)'')')
 WRITE(12,'(6X,''DIMENSION RP1(IDIM)'')')
 WRITE(12,'(6X,''DO 70023 I=1,IDIM'')')
 WRITE(12,'(''70023 READ(5,*)RP1(I)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
IF(FUN.EQ.'IMPMAT')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE IMPMAT(IP1,IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DO 70024 I=1,IDIM1'')')
 WRITE(12,'(''70024 WRITE(6,*)(IP1(I,J),J=1,IDIM2)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
 WRITE(12,'(6X,''SUBROUTINE IMPMAT(RP1,IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DO 70024 I=1,IDIM1'')')
 WRITE(12,'(''70024 WRITE(6,*)(RP1(I,J),J=1,IDIM2)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
```

```
IF(FUN.EQ.'LEEMAT')THEN
 IF(TIPO.EQ.'INT')THEN
 WRITE(12,'(6X,''SUBROUTINE LEEMAT(IP1,IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DIMENSION IP1(IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DO 70025 I=1,IDIM1'')')
 WRITE(12,'(''70025 READ(5,*)(IP1(I,J),J=1,IDIM2)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
 END IF
 WRITE(12,'(6X,''SUBROUTINE LEEMAT(RP1,IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DIMENSION RP1(IDIM1,IDIM2)'')')
 WRITE(12,'(6X,''DO 70025 I=1,IDIM1'')')
 WRITE(12,'(''70025 READ(5,*)(RP1(I,J),J=1,IDIM2)'')')
 WRITE(12,'(6X,''RETURN'',/,6X,''END'')')
 STA(IN,2)='1'
 RETURN
END IF
RETURN
END
```

* **PARTE 5:** Corridas de compilación y mapeo (Ligado).

KBB#SGCTES.COMPILA

```
@   PRT,S       KBB#SGCTES.COMPILA
@   PACK,S      KBB#SGCTES.
@   PSE         KBB#SGCTES.
@   FTN,FOSDR KBB#SGCTES.PREPRO,KBB#SGCREL.PREPRO
@   FTN,FOSDR KBB#SGCTES.WHILE,KBB#SGCREL.WHILE
@   FTN,FOSDR KBB#SGCTES.FOR,KBB#SGCREL.FOR
@   FTN,FOSDR KBB#SGCTES.REPEAT,KBB#SGCREL.REPEAT
@   FTN,FOSDR KBB#SGCTES.CASE,KBB#SGCREL.CASE
@   FTN,FOSDR KBB#SGCTES.VAR,KBB#SGCREL.VAR
@   FTN,FOSDR KBB#SGCTES.MAT,KBB#SGCREL.MAT
@   FTN,FOSDR KBB#SGCTES.VEC,KBB#SGCREL.VEC
@   FTN,FOSDR KBB#SGCTES.SEPARA,KBB#SGCREL.SEPARA
@   FTN,FOSDR KBB#SGCTES.DIMM,KBB#SGCREL.DIMM
@   FTN,FOSDR KBB#SGCTES.DIMV,KBB#SGCREL.DIMV
@   FTN,FOSDR KBB#SGCTES.PRIMI,KBB#SGCREL.PRIMI
@   FTN,FOSDR KBB#SGCTES.QUESOY,KBB#SGCREL.QUESOY
@   FTN,FOSDR KBB#SGCTES.FUNS,KBB#SGCREL.FUNS
@   FTN,FOSDR KBB#SGCTES.COMPAT,KBB#SGCREL.COMPAT
@   FTN,FOSDR KBB#SGCTES.SUBSTA,KBB#SGCREL.SUBSTA
@   FTN,FOSDR KBB#SGCTES.TABINI,KBB#SGCREL.TABINI
@   FTN,FOSDR KBB#SGCTES.BIBLIO,KBB#SGCREL.BIBLIO
@   FTN,FOSDR KBB#SGCTES.ARCHIV,KBB#SGCREL.ARCHIV
@   MASM,YS     KBB#SGCTES.OPT,KBB#SGCREL.OPT
@   MASM,YS     KBB#SGCTES.INFORM,KBB#SGCREL.INFORM
@   MASM,YS     KBB#SGCTES.CALIF,KBB#SGCREL.CALIF
@   PACK,S      KBB#SGCTES.
@   PREP        KBB#SGCREL.
@   MAP,EL      KBB#SGCTES.MAPEA,.MATFOR
@   PACK        KBB#SGCTES.
@   PSE         KBB#SGCTES.
T
@   PSE         KBB#SGCREL.
T
@   EOF
@ . FIN DE LA GENERACION DEL ABSOLUTO MATFOR
```

KBB#SGCTES.MAPEA

```
LIB KBB#SGCREL..                .
NOT TPF$.       .               .
    IN PREPRO   .               .
END     .       .               .
```

CAPITULO VI

"PRUEBAS"

..... **Archivo de entrada** .....

```
Prueba a la declaración de variables ***
Lineas de código con el objetivo de
probar lo anterior.


VAR
    A,B:INTEGER
    C,D:ENTERO
    HH,II,GGG :       MAT[1..5,1..8]   REAL
    VV,BB:VEC[1..10]ENTERO
    J,K : LOGICA
    HH1,II1,SSS1S: MAT[1..1000005,1..10]ENTERO
    VV1,BB1:VEC[1..10]REAL
    HH2*2,II2*4,XX*6: MAT[1..5,1..22222121CARACTER
    VV2*6,BB2*8:VEC[1..10]CHARACTER
END VAR
```

..... **Archivo de salida** .....

```
INTEGER A,B
INTEGER C,D
REAL HH,II,GGG
DIMENSION HH(5,8),II(5,8),GGG(5,8)
INTEGER VV,BB
DIMENSION VV(10),BB(10)
LOGICAL J,K
INTEGER HH1,II1,GGG1GG
DIMENSION HH1(1000005,10),II1(1000005,10),SSS1S(1000005,10)
REAL VV1,BB1
DIMENSION VV1(10),BB1(10)
CHARACTER HH2*2(5,2222212),II2*4(5,2222212),XX*6(5,2222212)
CHARACTER VV2*6(10),BB2*8(10)
```

```
                    ..... Archivo de entrada .....
                    Prueba para las estructuras: REPEAT,WHILE Y
                    FOR ###
                    Programa para encontrar la raiz cuadrada
                    de un número.


      VAR
         NUM:REAL
         RN,R : VEC[1..20] REAL
         TOLE: REAL
      ENDV
      TOLE=0.000001
      J=0
      FOR I=1 TO 20 DO
         BEGIN
         RN(I)=0.0
         R(I)=0.0
      END For
      REPEAT
      WRITE(6,'(2X,''DAME UN NUMERO'')')
      READ(5,'(F5.1)')NUM
      WRITE(6,'(7X,F5.1)')NUM
      IF(NUM.LT.0)THEN
        WRITE(6,'(2X,''ERROR EN ARGUMENTO'')')
      ELSE IF(NUM.NE.0.0)THEN
                RAIZ=1.0
                WHILE (ABS(NUM/(RAIZ**2)-1).GE.TOLE) DO
                RAIZ=(NUM/RAIZ+RAIZ)/2
                J=J+1
                RN(J)=NUM
                R(J)=RAIZ
      END IF
      UNT (NUM.EQ.0.0)
      FOR I=1 TO J DO
         BEGIN
         IF(RN(I).EQ.0.0)GO TO 100
         WRITE(6,'(''RAIZ C. DE:'',F5.1,''ES:'',F8.4)')RN(I),R(I)
      endf
100   WRITE(6,'(2X,''RAIZ C. DE:'',F5.1,'' ES:0.0'')')NUM
      WRITE(6,'(/,/,2X,''EN TOTAL METISTE'',I2,1X,''DATOS'')')I
      STOP
      END
```

..... **Archivo de salida** .....

```
      REAL NUM
      REAL RN,R
      DIMENSION RN(20),R(20)
      REAL TOLE
      TOLE=0.000001
      J=0
      DO 80000 I=1, 20
      RN(I)=0.0
      R(I)=0.0
80000 CONTINUE
80001 CONTINUE
      WRITE(6,'(2X,''DAME UN NUMERO'')')
      READ(5,'(F5.1)')NUM
      WRITE(6,'(7X,F5.1)')NUM
      IF(NUM.LT.0)THEN
      WRITE(6,'(2X,''ERROR EN ARGUMENTO'')')
      ELSE IF(NUM.NE.0.0)THEN
      RAIZ=1.0
80002 IF (.NOT.(ABS(NUM/(RAIZ**2)-1).GE.TOLE) ) GO TO 80003
      RAIZ=(NUM/RAIZ+RAIZ)/2
      GO TO 80002
80003 CONTINUE
      J=J+1
      RN(J)=NUM
      R(J)=RAIZ
      END IF
      IF(.NOT.(NUM.EQ.0.0))GO TO 80001
      DO 80004 I=1, J
      IF(RN(I).EQ.0.0)GO TO 100
      WRITE(6,'(''RAIZ C. DE:'',F5.1,''ES:'',F8.4)')RN(I),R(I)
80004 CONTINUE
  100 WRITE(6,'(2X,''RAIZ C. DE:'',F5.1,'' ES:0.0'')')NUM
      WRITE(6,'(/,/,2X,''EN TOTAL METISTE'',I2,1X,''DATOS'')')I
      STOP
      END
```

```
                    ..... Archivo de entrada .....
                    Prueba para las estructuras: REPEAT,WHILE \
                    FOR ***
                    Programa que ordena un vector.


      DIMENSION V(10)
      INTEGER V
      PRINT *,'*TU VECTOR ES DE 10 ELEMENTOS, DAME UNO POR UNO*'
      FOR NUM:=10 DOWNTO 1 DO
          READ *,V(NUM)
c     SE ORDENA EL VECTOR EN FORMA ASCENDENTE
      C=0
   15 FOR NUM:=1 TO 9 DO
       BEGIN
            IF(V(NUM).GT.V(NUM+1))THEN
               IAUX=V(NUM)
               V(NUM)=V(NUM+1)
               V(NUM+1)=IAUX
          ELSE IF(V(NUM).LE.V(NUM+1))THEN
                    C=C+1
                    IF(C.EQ.9)GO TO 20
                END IF
       END
      C=0
      GO TO 15
   20 PRINT *,'**VECTOR ORDENADO EN FORMA ASCENDENTE**'
      J=0
      REPEAT
        J=J+1
        WRITE(6,100)J,V(J)
      UNTIL (J.EQ.10)
c     OTRA MANERA DE MANDARLO A IMPRIMIR
      PRINT *,''**VECTOR ORDENADO EN FORMA DESCENDENTE**'
      J=0
      WHILE (J.GE.1) DO
         BEGIN
           WRITE(6,101)J,V(J)
           J=J-1
      END WHILE
  100 FORMAT(//,19X,'VEC(',I2,')=',I3)
  101 FORMAT(//,19X,'VEC(',I2,')=',I3)
      STOP
      END
```

..... **Archivo de salida** .....

```
      DIMENSION V(10)
      INTEGER V
      PRINT *,'*TU VECTOR ES DE 10 ELEMENTOS, DAME UNO POR UNO*'
      DO 80000 NUM=10,1,- 1
      READ *,V(NUM)
80000 CONTINUE
C      SE ORDENA EL VECTOR EN FORMA ASCENDENTE
      C=0
   15 FOR NUM!=1 TO 9 DO
      BEGIN
      IF(V(NUM).GT.V(NUM+1))THEN
      IAUX=V(NUM)
      V(NUM)=V(NUM+1)
      V(NUM+1)=IAUX
      ELSE IF(V(NUM).LE.V(NUM+1))THEN
      C=C+1
      IF(C.EQ.9)GO TO 20
      END IF
      END
      C=0
      GO TO 15
   20 PRINT *,'**VECTOR ORDENADO EN FORMA ASCENDENTE**'
      J=0
80001 CONTINUE
      J=J+1
      WRITE(6,100)J,V(J)
      IF(.NOT.(J.EQ.10)  )GO TO 80001
C      OTRA MANERA DE MANDARLO A IMPRIMIR
      PRINT *,''**VECTOR ORDENADO EN FORMA DESCENDENTE**'
      J=0
80002 IF (.NOT.(J.GE.1) ) GO TO 80003
      WRITE(6,101)J,V(J)
      J=J-1
      GO TO 80002
80003 CONTINUE
  100 FORMAT(//,19X,'VEC(',I2,')=',I3)
  101 FORMAT(//,19X,'VEC(',I2,')=',I3)
      STOP
      END
```

```
                ..... Archivo de entrada .....
                Prueba para las estructuras: REPEAT,FOR
                y WHILE en una subrutina ***
                Programa que reduce hasta el minimo término.


      character sigue*2
      integer nume,n,deno,d
      sigue = 'si'
      repeat
        for i=1 to 3 do
          begin
            read(5,100)nume,deno
            n=nume
            d=deno
            call minimo(nume,deno)
            write(6,101)n,d
            write(6,102)nume,deno
          end for
        write(6,'('' Quieres otro grupo de 3 ? '')')
        read(5,103)sigue
        write(6,'(12x,a2)')sigue
      until (sigue.eq.'no')
  100 format(i5,1x,i5)
  101 format(/,3x,'Datos de entrada : ',i5,'---',i5)
  102 format(/,3x,'Datos de salida  : ',i5,'---',i5)
  103 format(a2)
      stop
      end
      comienza subrutina para sacar el minimo
      subroutine minimo(nume,deno)
      integer nume,deno,ncopia,dcopia,residu
      ncopia = nume
      dcopia = deno
      while (dcopia.ne.0) do
        begin
          residu = mod(ncopia,dcopia)
          ncopia = dcopia
          dcopia = residu
      end while
      if(ncopia.gt.1)then
          nume = nume/ncopia
          deno = deno/ncopia
      end if
      return
      end
```

..... **Archivo de salida** .....

```
      CHARACTER SIGUE*2
      INTEGER NUME,N,DENO,D
      SIGUE = 'SI'
80000 CONTINUE
      DO 80001 I=1, 3
      READ(5,100)NUME,DENO
      N=NUME
      D=DENO
      CALL MINIMO(NUME,DENO)
      WRITE(6,101)N,D
      WRITE(6,102)NUME,DENO
80001 CONTINUE
      WRITE(6,'('' QUIERES OTRO GRUPO DE 3 ? '')')
      READ(5,103)SIGUE
      WRITE(6,'(12X,A2)')SIGUE
      IF(.NOT.(SIGUE.EQ.'NO')  )GO TO 80000
  100 FORMAT(I5,1X,I5)
  101 FORMAT(/,3X,'DATOS DE ENTRADA : ',I5,'---',I5)
  102 FORMAT(/,3X,'DATOS DE SALIDA  : ',I5,'---',I5)
  103 FORMAT(A2)
      STOP
      END
C      COMIENZA SUBRUTINA PARA SACAR EL MINIMO
      SUBROUTINE MINIMO(NUME,DENO)
      INTEGER NUME,DENO,NCOPIA,DCOPIA,RESIDU
      NCOPIA = NUME
      DCOPIA = DENO
80002 IF (.NOT.(DCOPIA.NE.0) ) GO TO 80003
      RESIDU = MOD(NCOPIA,DCOPIA)
      NCOPIA = DCOPIA
      DCOPIA = RESIDU
      GO TO 80002
80003 CONTINUE
      IF(NCOPIA.GT.1)THEN
      NUME = NUME/NCOPIA
      DENO = DENO/NCOPIA
      END IF
      RETURN
      END
```

```
.......... Archivo de entrada ..........
Prueba para la estructura: CASE $$$
Lineas de código con el objetivo de probar
la conversión del CASE.


character sig$5
integer a
iaaaa = 3
sig = 'ef'
ival = 1000
case iaaaa of
  1 : a=b$10
  2 : a=c$100
  3 : a=1205
  4 : a=125
  5 : a=ch/cd
  6 : a=2541
end case
print $,' Valor de a = ',a
case sig of
  'ab','cd','ef' : sig = 'ok'
  'jhij','klmn'  : sig = 'mal'
end c
print $,' Valor de sig = ',sig
case ival of
  10,20,30,40,50,60,70 : ilor = 1000
  100,200,300,400,500,600,700,800,900 : ilor = 5000
  101,201,301,401,501,601,701,801,901,1000 : ilor = 7000
endc
print $,' Valor de ilor = ',ilor
stop
end
```

..... Archivo de salida .....

```
 CHARACTER SIG*5
 INTEGER A
 IAAAA = 3
 SIG = 'EF'
 IVAL = 1000
 IF(IAAAA.EQ.1)THEN
 A=B*10
 ELSE IF(IAAAA.EQ.2)THEN
 A=C*100
 ELSE IF(IAAAA.EQ.3)THEN
 A=1205
 ELSE IF(IAAAA.EQ.4)THEN
 A=125
 ELSE IF(IAAAA.EQ.5)THEN
 A=CH/CD
 ELSE IF(IAAAA.EQ.6)THEN
 A=2541
 END IF
 PRINT *,' VALOR DE A = ',A
 IF(SIG.EQ.'AB'.OR.SIG.EQ.'CD'.OR.SIG.EQ.'EF')THEN
 SIG = 'OK'
 ELSE IF(SIG.EQ.'JHIJ'.OR.SIG.EQ.'KLMN')THEN
 SIG = 'MAL'
 END IF
 PRINT *,' VALOR DE SIG = ',SIG
 IF(IVAL.EQ.10.OR.IVAL.EQ.20.OR.IVAL.EQ.30.OR.
*IVAL.EQ.40.OR.IVAL.EQ.50.OR.IVAL.EQ.60.OR.
*IVAL.EQ.70)THEN
 ILOR = 1000
 ELSE IF(IVAL.EQ.100.OR.IVAL.EQ.200.OR.IVAL.EQ.300.OR.
*IVAL.EQ.400.OR.IVAL.EQ.500.OR.IVAL.EQ.600.OR.
*IVAL.EQ.700.OR.IVAL.EQ.800.OR.IVAL.EQ.900)THEN
 ILOR = 5000
 ELSE IF(IVAL.EQ.101.OR.IVAL.EQ.201.OR.IVAL.EQ.301.OR.
*IVAL.EQ.401.OR.IVAL.EQ.501.OR.IVAL.EQ.601.OR.
*IVAL.EQ.701.OR.IVAL.EQ.801.OR.IVAL.EQ.901.OR.
*IVAL.EQ.1000)THEN
 ILOR = 7000
 END IF
 PRINT *,' VALOR DE ILOR = ',ILOR
 STOP
 END
```

```
                ..... Archivo de entrada .....
                Prueba para las funciones:
                    Lee matriz.
                    Imprime matriz.
                    Suma matrices.
                    Lee vector.
                    Imprime vector.
                    suma vectores.
                    resta vectores. $$$
                Lineas de código con el objetivo de probar
                el funcionamiento de las funciones
                especificadas.


  VAR
    A : ENTERO
    SIG$2: CARACTER
    XX,YYYYY,ZZZZ : MAT(1..2,1..3) INTEGER
    MMM1 : MAT(1..3,1..2) INTEGER
    MMM : MAT(1..2,1..2) INTEGER
    V1,VV2,VVV3: VECT(1..2) ENTERO
  ENDV
2 WRITE(6,'('' QUIERES DAR DATOS DE TU MATRIZ?'')')
  READ(5,'(A2)')SIG
  IF(SIG.NE.'si')STOP
  print *,' leere a la matriz xx'
  ?XX
  &XX
  print *,' leere a la matriz yyyy'
  ?YYYYY
  &YYYYY
  print *,' estoy sumando zzzz=xx+yyyyy'
  ZZZZ=XX+YYYYY
  MMM=XX$MMM1
  print *,' estoy sumando yyyyy=xx+zzzz'
  YYYYY=XX+ZZZZ
  print *,' imprimire a YYYYY'
  &YYYYY
  A=A+1
  IF(A.LE.2)GO TO 2
  WRITE(6,'('' SOLO DOS VECES ...ADIOS'')')
  print *,' leere al vector v1'
  ?V1
  &V1
  print *,' leere al vector vv2'
  ?VV2
  &VV2
  print *,' sumare los vec vvv3=v1+vv2'
  VVV3=V1+VV2
  &VVV3
  print *,' sumare los vec v1=VVV3+vv2'
  V1=VVV3+VV2
  print *,' imprimire el vector V1'
  &V1
```

```
print *,' restare vvv3=v1-vv2'
VVV3=V1-VV2
print *,' imprimire el vector VVV3'
&VVV3
STOP
END
```

..... Archivo de salida .....

```
 INTEGER A
 CHARACTER SIG*2
 INTEGER XX,YYYYY,ZZZZ
 DIMENSION XX(2,3),YYYYY(2,3),ZZZZ(2,3)
 INTEGER MMM1
 DIMENSION MMM1(3,2)
 INTEGER MMM
 DIMENSION MMM(2,2)
 INTEGER V1,VV2,VVV3
 DIMENSION V1(2),VV2(2),VVV3(2)
2 WRITE(6,'('' QUIERES DAR DATOS DE TU MATRIZ?'')')
 READ(5,'(A2)')SIG
 IF(SIG.NE.'SI')STOP
 PRINT *,' LEERE A LA MATRIZ XX'
 ID$1=2
 ID$2=3
 CALL LEEMAT(XX,ID$1,ID$2)
 ID$1=2
 ID$2=3
 CALL IMPMAT(XX,ID$1,ID$2)
 PRINT *,' LEERE A LA MATRIZ YYYY'
 ID$1=2
 ID$2=3
 CALL LEEMAT(YYYYY,ID$1,ID$2)
 ID$1=2
 ID$2=3
 CALL IMPMAT(YYYYY,ID$1,ID$2)
 PRINT *,' ESTOY SUMANDO ZZZZ=XX+YYYYY'
 ID$1=2
 ID$2=3
 CALL SUMMAT(ZZZZ,XX,YYYYY,ID$1,ID$2)
 ID$1=2
 ID$2=2
 ID$3=3
 CALL MULMAT(MMM,XX,MMM1,ID$1,ID$2,ID$3)
 PRINT *,' ESTOY SUMANDO YYYYY=XX+ZZZZ'
 ID$1=2
 ID$2=3
 CALL SUMMAT(YYYYY,XX,ZZZZ,ID$1,ID$2)
 PRINT *,' IMPRIMIRE A YYYYY'
 ID$1=2
 ID$2=3
 CALL IMPMAT(YYYYY,ID$1,ID$2)
```

```
      A=A+1
      IF(A.LE.2)GO TO 2
      WRITE(6,'(''   SOLO DOS VECES ...ADIOS'')')
      PRINT *,' LEERE AL VECTOR V1'
      ID$1=2
      CALL LEEVEC(V1,ID$1)
      ID$1=2
      CALL IMPVEC(V1,ID$1)
      PRINT *,' LEERE AL VECTOR VV2'
      ID$1=2
      CALL LEEVEC(VV2,ID$1)
      ID$1=2
      CALL IMPVEC(VV2,ID$1)
      PRINT *,' SUMARE LOS VEC VVV3=V1+VV2'
      ID$1=2
      CALL SUMVEC(VVV3,V1,VV2,ID$1)
      ID$1=2
      CALL IMPVEC(VVV3,ID$1)
      PRINT *,' SUMARE LOS VEC V1=VVV3+VV2'
      ID$1=2
      CALL SUMVEC(V1,VVV3,VV2,ID$1)
      PRINT *,' IMPRIMIRE EL VECTOR V1'
      ID$1=2
      CALL IMPVEC(V1,ID$1)
      PRINT *,' RESTARE VVV3=V1-VV2'
      ID$1=2
      CALL RESVEC(VVV3,V1,VV2,ID$1)
      PRINT *,' IMPRIMIRE EL VECTOR VVV3'
      ID$1=2
      CALL IMPVEC(VVV3,ID$1)
      STOP
      END
      SUBROUTINE LEEMAT(IP1,IDIM1,IDIM2)
      DIMENSION IP1(IDIM1,IDIM2)
      DO 70025 I=1,IDIM1
70025 READ(5,*)(IP1(I,J),J=1,IDIM2)
      RETURN
      END
      SUBROUTINE IMPMAT(IP1,IDIM1,IDIM2)
      DIMENSION IP1(IDIM1,IDIM2)
      DO 70024 I=1,IDIM1
70024 WRITE(6,*)(IP1(I,J),J=1,IDIM2)
      RETURN
      END
      SUBROUTINE SUMMAT(IP1,IP2,IP3,IDIM1,IDIM2)
      DIMENSION  IP1(IDIM1,IDIM2),IP2(IDIM1,IDIM2),
     *IP3(IDIM1,IDIM2)
      DO 70009 I=1,IDIM1
      DO 70009 J=1,IDIM2
70009 IP1(I,J)=IP2(I,J)+IP3(I,J)
      RETURN
      END
```

```
          ..... Archivo de entrada .....
          Prueba al CASE con el llamado interno a
          funciones de MATfor. ***
          Lineas de código con el objetivo de probar
          lo especificado anteriormente.

character sig*5
integer a,z
ARRAY
    UNO,DOS,TRES:VEC(1..3) ENTEROS
    MAT1,MAT2: MAT[1..2,1..3] REALES
ENDA
iaaaa=3
sig='EF'
ival=1000
b=10
c=20
case iaaaa of
123456789 :

      a=b*10
  2 :
     a=c*100
  3 :
      a=1205
  4 :
      a=125
  5 :
      a=ival/2
  6 :
      a=2541
end case
PRINT *,'VALOR DE A = ',a
CASE sig OF
  'AB3456789','CD','EF' : sig='ok'
  'JHIJ','KLMN': sig='MAL'
END C
PRINT *,'VALOR DE SIG = ',sig
CASE ival  OF
 10,20,30,40,50,60,70 : ILOR = 1000
 100,200,300,400,500,600,700,800,900 : ILOR = 5000
 101,201,301,401,501,601,701,801,901,1000 : ILOR = 7000
ENDC
PRINT *,'VALOR DE ILOR = ',ILOR
ipru=2
j=10
case ipru  OF
  1 :
      begin
        icon1=10
        icon2=20
        icon3=30
```

```
            print *,' valor de icon1=',icon1,' de icon2=',icon2
     c,' de icon3= ',icon3
            end begin
        2 :
            begin
              repeat
                z=z+1
              until (z.eq.5)
              isal1=20*2
+ esta     esta  es una linea de comentario
              isal2=isal1/2
              write(6,*)' valor de z= ',z,' de isal1=',isal1
            end b
        3 : begin
             while (j.ge.1) do
                begin
                  write(6,'(' valor de j en while = ',i2)')j
                  j=j-1
                end while
              iola1=20*2
              iola2=isal1/2
              print *,' valor de j= ',j
            endb
      END CASE
      ivec=1
      case ivec of
      1,2,3 : begin
 c este es un comentario para lectura de vectores
              write(*,*)'  dame el vector uno(3)'
               ?uno
              print *,'  dame el vector dos(3)'
               ?dos
              write(*,*)' dame el vector tres(3)'
               ?tres
               &uno
               &dos
               &tres
              endb
c este es un com. para lectura de matrices
      4,5,6 : begin
              write(*,*)' dame la matriz mat1(2,3)'
              ?mat1
              write(*,*)' dame la matriz mat2(2,3)'
              ?mat2
              &mat1
              &mat2
              end begin
      end case
      STOP
      END
```

```
      CHARACTER SIG*5
      INTEGER A,Z
      INTEGER UNO,DOS,TRES
      DIMENSION UNO(3),DOS(3),TRES(3)
      REAL MAT1,MAT2
      DIMENSION MAT1(2,3),MAT2(2,3)
      IAAAA=3
      SIG='EF'
      IVAL=1000
      B=10
      C=20
      IF(IAAAA.EQ.123456789)THEN
      A=B*10
      ELSE IF(IAAAA.EQ.2)THEN
      A=C*100
      ELSE IF(IAAAA.EQ.3)THEN
      A=1205
      ELSE IF(IAAAA.EQ.4)THEN
      A=125
      ELSE IF(IAAAA.EQ.5)THEN
      A=IVAL/2
      ELSE IF(IAAAA.EQ.6)THEN
      A=2541
      END IF
      PRINT *,'VALOR DE A = ',A
      IF(SIG.EQ.'AB3456789'.OR.SIG.EQ.'CD'.OR.SIG.EQ.'EF')THEN
      SIG='OK'
      ELSE IF(SIG.EQ.'JHIJ'.OR.SIG.EQ.'KLMN')THEN
      SIG='MAL'
      END IF
      PRINT *,'VALOR DE SIG = ',SIG
      IF(IVAL.EQ.10.OR.IVAL.EQ.20.OR.IVAL.EQ.30.OR.
     *IVAL.EQ.40.OR.IVAL.EQ.50.OR.IVAL.EQ.60.OR.
     *IVAL.EQ.70)THEN
      ILOR = 1000
      ELSE IF(IVAL.EQ.100.OR.IVAL.EQ.200.OR.IVAL.EQ.300.OR.
     *IVAL.EQ.400.OR.IVAL.EQ.500.OR.IVAL.EQ.600.OR.
     *IVAL.EQ.700.OR.IVAL.EQ.800.OR.IVAL.EQ.900)THEN
      ILOR = 5000
      ELSE IF(IVAL.EQ.101.OR.IVAL.EQ.201.OR.IVAL.EQ.301.OR.
     *IVAL.EQ.401.OR.IVAL.EQ.501.OR.IVAL.EQ.601.OR.
     *IVAL.EQ.701.OR.IVAL.EQ.801.OR.IVAL.EQ.901.OR.
     *IVAL.EQ.1000)THEN
      ILOR = 7000
      END IF
      PRINT *,'VALOR DE ILOR = ',ILOR
      IPRU=2
      J=10
      IF(IPRU.EQ.1)THEN
      ICON1=10
      ICON2=20
```

```
      ICON3=30
      PRINT *,' VALOR DE ICON1=',ICON1,' DE ICON2=',ICON2
     *,' DE ICON3= ',ICON3
      ELSE IF(IPRU.EQ.2)THEN
80000 CONTINUE
      Z=Z+1
      IF(.NOT.(Z.EQ.5)  )GO TO 80000
      ISAL1=20*2
C ESTA   ESTA  ES UNA LINEA DE COMENTARIO
      ISAL2=ISAL1/2
      WRITE(6,*)' VALOR DE Z= ',Z,' DE ISAL1=',ISAL1
      ELSE IF(IPRU.EQ.3)THEN
80001 IF (.NOT.(J.GE.1) ) GO TO 80002
      WRITE(6,'('' VALOR DE J EN WHILE = ',I2)')J
      J=J-1
      GO TO 80001
80002 CONTINUE
      IOLA1=20*2
      IOLA2=ISAL1/2
      PRINT *,' VALOR DE J= ',J
      END IF
      IVEC=1
      IF(IVEC.EQ.1.OR.IVEC.EQ.2.OR.IVEC.EQ.3)THEN
C ESTE ES UN COMENTARIO PARA LECTURA DE VECTORES
      WRITE(*,*)'  DAME EL VECTOR UNO(3)'
      ID$1=3
      CALL LEEVEC(UNO,ID$1)
      PRINT *,'  DAME EL VECTOR DOS(3)'
      ID$1=3
      CALL LEEVEC(DOS,ID$1)
      WRITE(*,*)' DAME EL VECTOR TRES(3)'
      ID$1=3
      CALL LEEVEC(TRES,ID$1)
      ID$1=3
      CALL IMPVEC(UNO,ID$1)
      ID$1=3
      CALL IMPVEC(DOS,ID$1)
      ID$1=3
      CALL IMPVEC(TRES,ID$1)
C ESTE ES UN COM. PARA LECTURA DE MATRICES
      ELSE IF(IVEC.EQ.4.OR.IVEC.EQ.5.OR.IVEC.EQ.6)THEN
      WRITE(*,*)' DAME LA MATRIZ MAT1(2,3)'
      ID$1=2
      ID$2=3
      CALL LEEMAT(MAT1,ID$1,ID$2)
      WRITE(*,*)' DAME LA MATRIZ MAT1(2,3)'
      ID$1=2
      ID$2=3
      CALL LEEMAT(MAT2,ID$1,ID$2)
      ID$1=2
      ID$2=3
      CALL IMPMAT(MAT1,ID$1,ID$2)
```

```
      ID$1=2
      ID$2=3
      CALL IMPMAT(MAT2,ID$1,ID$2)
      END IF
      STOP
      END
      SUBROUTINE LEEVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      READ(5,*)IP1
      RETURN
      END
      SUBROUTINE IMPVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      WRITE(6,*)IP1
      RETURN
      END
      SUBROUTINE LEEMAT(RP1,IDIM1,IDIM2)
      DIMENSION RP1(IDIM1,IDIM2)
      DO 70025 I=1,IDIM1
70025 READ(5,*)(RP1(I,J),J=1,IDIM2)
      RETURN
      END
      SUBROUTINE IMPMAT(RP1,IDIM1,IDIM2)
      DIMENSION RP1(IDIM1,IDIM2)
      DO 70024 I=1,IDIM1
70024 WRITE(6,*)(RP1(I,J),J=1,IDIM2)
      RETURN
      END
```

..... **Archivo de entrada** .....

```
Prueba al caracter de reconocimiento de
funcion (!) acompañado de una etiqueta. ***

      var
          a : entera
          hola,qui : mat[1..2,1..4)enteras
          UNO,DOS,tres :VEC [1..2]entero
          mat1,mat2 : mat(1..3,1..2) entera
          mat3,mat4 : mat(1..2,1..5) entera
          vec1,vec2 :vec [1..4] entera
      END var
 101   a=a+1
      if(a.le.2)go to 200
      go to 110
 !200  ?uno
      go to 101
 !110    &uno
  !    ?mat1
  !    ?mat2
  !    mat1    =    mat1   +   mat2
  !    &mat1
       stop
       end
```

..... **Archivo de salida** .....

```
      INTEGER A
      INTEGER HOLA,QUI
      DIMENSION HOLA(2,4),QUI(2,4)
      INTEGER UNO,DOS,TRES
      DIMENSION UNO(2),DOS(2),TRES(2)
      INTEGER MAT1,MAT2
      DIMENSION MAT1(3,2),MAT2(3,2)
      INTEGER MAT3,MAT4
      DIMENSION MAT3(2,5),MAT4(2,5)
      INTEGER VEC1,VEC2
      DIMENSION VEC1(4),VEC2(4)
 101  A=A+1
      IF(A.LE.2)GO TO 200
      GO TO 110
200   ID$1=2
      CALL LEEVEC(UNO,ID$1)
      GO TO 101
110   ID$1=2
      CALL IMPVEC(UNO,ID$1)
      ID$1=3
      ID$2=2
      CALL LEEMAT(MAT1,ID$1,ID$2)
```

```
      ID$1=3
      ID$2=2
      CALL LEEMAT(MAT2,ID$1,ID$2)
      ID$1=3
      ID$2=2
      CALL SUMMAT(MAT1,MAT1,MAT2,ID$1,ID$2)
      ID$1=3
      ID$2=2
      CALL IMPMAT(MAT1,ID$1,ID$2)
      STOP
      END
      SUBROUTINE LEEVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      READ(5,*)IP1
      RETURN
      END
      SUBROUTINE IMPVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      WRITE(6,*)IP1
      RETURN
      END
      SUBROUTINE LEEMAT(IP1,IDIM1,IDIM2)
      DIMENSION IP1(IDIM1,IDIM2)
      DO 70025 I=1,IDIM1
70025 READ(5,*)(IP1(I,J),J=1,IDIM2)
      RETURN
      END
      SUBROUTINE SUMMAT(IP1,IP2,IP3,IDIM1,IDIM2)
      DIMENSION IP1(IDIM1,IDIM2),IP2(IDIM1,IDIM2),
     *IP3(IDIM1,IDIM2)
      DO 70009 I=1,IDIM1
      DO 70009 J=1,IDIM2
70009 IP1(I,J)=IP2(I,J)+IP3(I,J)
      RETURN
      END
      SUBROUTINE IMPMAT(IP1,IDIM1,IDIM2)
      DIMENSION IP1(IDIM1,IDIM2)
      DO 70024 I=1,IDIM1
70024 WRITE(6,*)(IP1(I,J),J=1,IDIM2)
      RETURN
      END
```

```
                    ..... Archivo de entrada .....
                    Prueba al CASE con el llamado interno a
                    funciones de MATfor,con líneas en blanco,
                    líneas de comentarios y líneas de
                    continuación.  $$$
                    Líneas de código con el objetivo de probar
                    lo especificado anteriormente.


      ARRAY
          xx,UNO,DOS,TRES:VEC(1..3) ENTEROS
          matriz,MAT2: MAT[1..2,1..3] REALES
      end array
      ipru=2
      j=10
      case ipru  of
        1 :
              begin

              icon1=10
              icon2=20
              icon3=30
              print $,' valor de icon1=',icon1,' de icon2=',icon2
    c,' de icon3= ',icon3
             end begin
        2:
             begin
               repeat
                 z=z+1
               until (z.eq.5)
               isal1=20$2
+ esta    esta  es una linea de comentario
               isal2=isal1/2
               write(6,$)' valor de z= ',z,' de isal1=',isal1
             end b
        3 : begin
              while (j.ge.1) do
                 begin
                   write(6,'('' valor de j en while ='',i2)')j
                   j=j-1
                 end w
               iolal=20$2
               iola2=isal1/2
               print $,' valor de j= ',j
             endb
      END CASE
      ivec=1
```

```
     case ivec of
     1,2,3 : begin
 c este es un comentario para lectura de vectores
               ?uno
               ?dos
               ?tres
               &uno
               &dos
               &tres
              endb
c este es un com. para lectura de matrices
     4,5,6 : begin
               ?matriz
               ?mat2
               &matriz
               &mat2
               end begin
      end case
      ia=2
      while (ia.eq.2) do
       begin
         ?xx
         case xx(1) of
           1 : print *,' valor del xx(1) =',xx(1)
           2 : begin
                xaux=xx(1)
                xx(1)=xx(2)
                xx(2)=xaux
               end begin
          end case
         ?matriZ
         case matriz(1,1) of
           1 : print *,' valor del matriz(1,1) =',matriz(1,1)
           2 : begin
                maux=matriz(1,1)
                matriz(1,1)=matriz(1,2)
                matriz(1,2)=maux
               end begin
          end case
        ia=ia+1
      end while
      STOP
      END
```

```
      INTEGER XX,UNO,DOS,TRES
      DIMENSION XX(3),UNO(3),DOS(3),TRES(3)
      REAL MATRIZ,MAT2
      DIMENSION MATRIZ(2,3),MAT2(2,3)
      IPRU=2
      J=10
      IF(IPRU.EQ.1)THEN
      ICON1=10
      ICON2=20
      ICON3=30
      PRINT *,' VALOR DE ICON1=',ICON1,' DE ICON2=',ICON2
     *,' DE ICON3= ',ICON3
      ELSE IF(IPRU.EQ.2)THEN
80000 CONTINUE
      Z=Z+1
      IF(.NOT.(Z.EQ.5)  )GO TO 80000
      ISAL1=20*2
C ESTA   ESTA  ES UNA LINEA DE COMENTARIO
      ISAL2=ISAL1/2
      WRITE(6,*)' VALOR DE Z= ',Z,' DE ISAL1=',ISAL1
      ELSE IF(IPRU.EQ.3)THEN
80001 IF (.NOT.(J.GE.1) ) GO TO 80002
      WRITE(6,'('' VALOR DE J EN WHILE ='',I2)')J
      J=J-1
      GO TO 80001
80002 CONTINUE
      IOLA1=20*2
      IOLA2=ISAL1/2
      PRINT *,' VALOR DE J= ',J
      END IF
      IVEC=1
      IF(IVEC.EQ.1.OR.IVEC.EQ.2.OR.IVEC.EQ.3)THEN
C ESTE ES UN COMENTARIO PARA LECTURA DE VECTORES
      ID$1=3
      CALL LEEVEC(UNO,ID$1)
      ID$1=3
      CALL LEEVEC(DOS,ID$1)
      ID$1=3
      CALL LEEVEC(TRES,ID$1)
      ID$1=3
      CALL IMPVEC(UNO,ID$1)
      ID$1=3
      CALL IMPVEC(DOS,ID$1)
      ID$1=3
      CALL IMPVEC(TRES,ID$1)
C ESTE ES UN COM. PARA LECTURA DE MATRICES
      ELSE IF(IVEC.EQ.4.OR.IVEC.EQ.5.OR.IVEC.EQ.6)THEN
      ID$1=2
      ID$2=3
      CALL LEEMAT(MATRIZ,ID$1,ID$2)
```

```
      ID$1=2
      ID$2=3
      CALL LEEMAT(MAT2,ID$1,ID$2)
      ID$1=2
      ID$2=3
      CALL IMPMAT(MATRIZ,ID$1,ID$2)
      ID$1=2
      ID$2=3
      CALL IMPMAT(MAT2,ID$1,ID$2)
      END IF
      IA=2
80003 IF (.NOT.(IA.EQ.2) ) GO TO 80004
      ID$1=3
      CALL LEEVEC(XX,ID$1)
      IF(XX(1).EQ.1)THEN
      PRINT *,' VALOR DEL XX(1) =',XX(1)
      ELSE IF(XX(1).EQ.2)THEN
      XAUX=XX(1)
      XX(1)=XX(2)
      XX(2)=XAUX
      END IF
      ID$1=2
      ID$2=3
      CALL LEEMAT(MATRIZ,ID$1,ID$2)
      IF(MATRIZ(1,1).EQ.1)THEN
      PRINT *,' VALOR DEL MATRIZ(1,1) =',MATRIZ(1,1)
      ELSE IF(MATRIZ(1,1).EQ.2)THEN
      MAUX=MATRIZ(1,1)
      MATRIZ(1,1)=MATRIZ(1,2)
      MATRIZ(1,2)=MAUX
      END IF
      IA=IA+1
      GO TO 80003
80004 CONTINUE
      STOP
      END
      SUBROUTINE LEEVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      READ(5,*)IP1
      RETURN
      END
      SUBROUTINE IMPVEC(IP1,IDIM)
      DIMENSION IP1(IDIM)
      WRITE(6,*)IP1
      RETURN
      END
      SUBROUTINE LEEMAT(RP1,IDIM1,IDIM2)
      DIMENSION RP1(IDIM1,IDIM2)
      DO 70025 I=1,IDIM1
70025 READ(5,*)(RP1(I,J),J=1,IDIM2)
      RETURN
      END
```

```
      SUBROUTINE  IMPMAT(RP1, IDIM1, IDIM2)
      DIMENSION RP1(IDIM1,IDIM2)
      DO 70024 I=1,IDIM1
70024 WRITE(6,*)(RP1(I,J),J=1,IDIM2)
      RETURN
      END
```

CAPITULO VII

"PROCEDIMIENTO DE EJECUCION"

En este capítulo se verán: la forma de ejecutar el preprocesador, manera de obtener un programa FORtran, equivalente del seudolenguaje MATfor y las alternativas permitidas en el uso del preprocesador.

VII.1 FORMATO Y ALTERNATIVAS

La sintaxis del llamado al preprocesador es la siguiente:

>@MATFOR,opt Archivo-entrada.,Archivo-salida.elemento

donde

opt: Corresponde a las opciones posibles de MATfor.
SYM--> Para la obtención de un programa fuente (simbólico). Conversión del programa en MATfor a FORtran.

ABS--> Para la obtención de un programa ejecutable (absoluto).

Archivo-entrada: Nombre del archivo de entrada (archivo de datos), que corresponde al programa en MATfor. Debe de cumplir con los requisitos que exige el sistema para la creación de un archivo.

Calif*nombre-archivo.

Calif--> Calificador o cuenta del usuario (identificador de su subdirección, gerencia, división y departamento).

* --> Caracter asterisco.

nombre-archivo--> Los tres primeros caracteres corresponden a las iniciales del usuario y los restantes, hasta un máximo de 12 caracteres, a lo deseado por él mismo.

Archivo-salida.elemento: Nombre del elemento que pertenece o pertenecerá al archivo de salida (archivo de programas ya existente). En otros términos, el nombre de un archivo que pertenece a "x" directorio.

En el elemento indicado que puede existir o no, quedará el prog. simbólico o el prog. ejecutable, según la opción con la cual se ejecutó el preprocesador.

El elemento debe de cumplir con :

Calif$nombre-archivo.nombre-elemento

nombre-elemento--> Cualquier nombre, puede empezar con letra o número, pero no debe de exceder de 12 caracteres.

VII.2 OBTENCION DE UN PROGRAMA FUENTE

Es necesario aclarar ciertos conceptos de archivos y directorios. Aquí se está manejando un Archivo de datos como aquel que es independiente de cualquier directorio y un archivo de programas, valga la redundancia, aquél que contiene programas y se les llama elementos de dicho archivo, esto es semejante en micros a decir el archivo de "x" directorio, o sea que un directorio es un archivo de programas.

Un programa fuente es un programa simbólico y éste se obtiene ejecutando al preprocesador con la opción "SYM".

Para entender este procedimiento vamos a suponer que en el archivo de datos "KBB$SGCPRUEBA." tenemos nuestro programa en MATfor y que en el archivo de programas "KBB$SGCPROG." vamos a dejar la conversión, para esto se creará un elemento (programa) dentro de dicho archivo de nombre "SIMBOLICO1" que tendrá el programa en FORtran.

```
>@MATFOR,SYM KBB$SGCPRUEBA.,KBB$SGCPROG.SIMBOLICO1
```

El preprocesador toma por default el calificador del usuario que está en sesión; por lo tanto, si los calificadores de los archivos a utilizar (de datos y de programas) corresponden al usuario que está ejecutando se pueden omitir.

La ejecución quedaria de la siguiente manera:

```
>@MATFOR,SYM $GCPRUEBA.,$GCPROG.SIMBOLICO1
```

VII.3 OBTENCION DE UN PROGRAMA EJECUTABLE

Un programa ejecutable es un absoluto, y este programa no es editable. En el sistema Univac 1100/80 la forma de ejecutar un absoluto es de la siguiente manera:

```
>@XQT nombre-del-absoluto.
```

El "@XQT" es la tarjeta de control que le indica al sistema que se trata de un programa absoluto.

Para obtener a través de nuestro preprocesador un programa absoluto, supongamos que tenemos nuestro anterior archivo de datos "KBB*SGCPRUEBA.", y el absoluto correspondiente lo deseamos dejar en un elemento del archivo de programas "KBB*SGCPROG.", con nombre "ABSOLUTO1".

```
>@MATFOR,ABS KBB*SGCPRUEBA.,KBB*SGCPROG.ABSOLUTO1
```

Si el calificador del usuario en sesión es el mismo que el de los archivos, entonces se puede ejecutar de la siguiente manera:

```
>@MATFOR,ABS SGCPRUEBA.,SGCPROG.ABSOLUTO1
```

Si se verifica qué elementos se tienen en el archivo de programas, se observará que existen dos más. El simbólico1 que es un programa fuente, y el absoluto1 que es el ejecutable.

La forma de hacerlo en la máquina Univac 1100/80 es a través de la tarjeta de control "@PRT,T archivo-de-programas."

```
>@PRT,T KBB*SGCPROG.
   .          .
   .          .
   .          .
>SYM       SIMBOLICO1
>ABS       ABSOLUTO1
```

Tipo de elemento — nombre del elemento

VII.4 NOMBRES POR DEFAULT A LOS PROGRAMAS FUENTE Y EJECUTABLE

Ya sabemos como obtener un programa fuente y uno ejecutable a través de la sintaxis general. Ahora veremos las alternativas de ejecutar al preprocesador y los nombres por default que se toman.

Si ejecutamos de la siguiente manera :

```
>@MATFOR,SYM KBB$SGCPRUEBA.,KBB$SGCPROG.
```

El nombre del elemento donde queda el simbólico es "MATFOR". Es equivalente a haber puesto "KBB$SGCPROG.MATFOR" en el campo del Archivo-salida.elemento .

Si se ejecuta de esta forma :

```
>@MATFOR,SYM KBB$SGCPRUEBA.
```

La conversión de dicho programa queda en un archivo de datos temporal (será borrado cuando se termine la sesión) de nombre "MATFOR", toma el calificador del usuario en sesión y crea el archivo "CALIF$MATFOR".

Si se ejecuta de la siguiente manera :

```
>@MATFOR,ABS KBB$SGCPRUEBA.,KBB$SGCPROG.
```

El absoluto queda con el nombre "MATFOR" dentro del archivo de programas que se encuentra en el campo perteneciente al archivo-salida.elemento .

NOTA: El nombre por default para el programa simbólico y absoluto es el mismo. Estos elementos son independientes, no causan ningún problema ya que uno es el fuente y el otro el ejecutable.

Si volviera a ejecutar otra vez con otro archivo de entrada, y el mismo de salida, y no especifico el elemento, perdería el programa fuente o ejecutable anterior porque será sustituido por el nuevo.

Si ahora se ejecuta de esta manera :

```
>@MATFOR,ABS KBB$SGCPRUEBA.
```

El programa ejecutable queda en el elemento de un archivo temporal y es "TPF$.NAME$" .

Si estos archivos temporales dados por default, resultan del interés del usuario, se recomienda que se copien a archivos permanentes. De esta manera se evita procesar de nuevo.

$ ARCHIVOS QUE CREA Y SE ASIGNA EL PREPROCESADOR CUANDO SE LE LLAMA

```
>@MATFOR,SYM KBB$SGCPRUEBA.,KBB$SGCPROG.ARCHIVO1
```

KBB$MATFOR.	Conversión (simbólico) .
KBB$SYM.	Archivo de datos que contiene un sólo registro y es la tarjeta de control generada para copiar MATFOR a SGCPROG.ARCHIVO1
KBB$SGCPROG.ARCHIVO1	Conversión (simbólico) .

```
>@MATFOR,ABS KBB$SGCPRUEBA.,KBB$SGCPROG.ARCHIVO1
```

KBB$MATFOR.	Conversión (simbólico) .
KBB$FOREXT.	Archivo de datos que contiene la compilación de la conversión y genera el absoluto dejándolo en SGCPROG.ARCHIVO1 .
KBB$SGCPROG.ARCHIVO1	Programa absoluto no editable, sólo ejecutable.

>@MATFOR,SYM KBB$SGCPRUEBA.,KBB$SGCPROG.

KBB$MATFOR.	Conversión (simbólico) .
KBB$SYM.	Archivo de datos que contiene un sólo registro y es la tarjeta de control generada para copiar MATFOR a SGCPROG.MATFOR.
KBB$SGCPROG.MATFOR	Conversión (simbólico) .

>@MATFOR,ABS KBB$SGCPRUEBA.,KBB$SGCPROG.

KBB$MATFOR.	Conversión (simbólico) .
KBB$FOREXT.	Archivo de datos que contiene la compilación de la conversión y genera el absoluto dejándolo en SGCPROG.MATFOR .
KBB$SGCPROG.MATFOR	Programa absoluto no editable, sólo ejecutable.

NOTA: Hay que tener cuidado en no olvidar ponerle el punto al archivo de salida.

* Si se olvidara ponerlo y se ejecuta con la opción "SYM", la conversión queda en el archivo de datos temporal "MATFOR.", la tarjeta del copiado en el archivo de datos temporal "SYM.", pero se encuentra mal generada porque no se le puso punto al archivo de salida.

* Si con la opción "ABS" se olvidara también el punto al archivo de salida, la conversión queda en "MATFOR.", la compilación en "FOREXT.",ésta compilación se logró y el absoluto queda en "TPF$.NAME$", lo que estuvo mal fue la tarjeta generada para copiar este absoluto al deseado o al default.

En estos casos es conveniente ejecutar nuevamente, si es que no se tienen bien presentes estos archivos o, en su defecto, hacer los copiados por fuera a los simbólicos o absolutos deseados.

```
>@MATFOR,SYM KBB$BGCPRUEBA.
```

KBB$MATFOR.	Conversión (simbólico) .

```
>@MATFOR,ABS KBB$BGCPRUEBA.
```

KBB$MATFOR.	Conversión (simbólico) .
KBB$FOREXT.	Archivo de datos que contiene la compilación de la conversión y genera el absoluto dejándolo en TPF$.NAME$
KBBTPF.NAME$	Programa absoluto no editable, sólo ejecutable.

Si no se le pone punto al archivo de entrada, el preprocesador lo detecta y manda un mensaje de error.
Por ejemplo :

```
>@MATFOR,SYM KBB$BGCPRUEBA

MATFOR SGC/RHR VER.01 MM/DD/AA-HH:MM:SS
ERROR!!... EN EL LLAMADO DEL PREPROCESADOR
(FALTO EL PUNTO EN EL ARCHIVO DE ENTRADA)
```

La linea de identificación del preprocesador, que es la primera que aparece cuando se le llama a MATFOR, tiene el siguiente significado :

1) MATFOR : Nombre del preprocesador.

2) SGC/RHR : Iniciales que corresponde a:

   SGC : Sergio García Corona (Pasante).
   RHR : Raymundo Hugo Rangel (Asesor).

3) VER.01 : Versión número 1.

4) MM/DD/AA-HH:MM:SS : Corresponden al mes, día y año; hora, minutos y segundos, en que se ejecutó el preprocesador "MATFOR" .

CONCLUSIONES

El preprocesador MATfor integra las estructuras de control de la programación estructurada y el manejo a alto nivel de vectores y matrices.

MATfor resulta de gran apoyo para los programadores y en empresas que tienen y desarrollan aplicaciones en FORtran. Los beneficios en el desarrollo de un programa en MATfor se reflejan en una mayor productividad (puesto que se reduce el tiempo de codificación), confiabilidad en los resultados (ya que evita la tendencia a cometer errores), facilita el mantenimiento de los programas y reduce la violación a las reglas de la programación estructurada.

El preprocesador MATfor no intenta hacer un nuevo lenguaje de programación, su principal objetivo es que el usuario tenga instrucciones adicionales a las que ya posee en FORtran, para la mejor realización de sus tareas.

MATfor puede no detectar una línea de código que se encuentre mala al traducirla, pero el compilador FORtran es perfectamente capaz de detectar cualquier error de sintaxis que pudiera pasársele al preprocesador.

Una ventaja de un preprocesador es que no se tiene que escribir un compilador para obtener un lenguaje mejor.

El objetivo de un preprocesador es el de hacer al lenguaje mejor, MATfor se apegó a este objetivo e hizo a FORtran un lenguaje de programación mejor.

Matfor puede aumentar esta mejora y crecer en sus características, puesto que durante su desarrollo nacieron ideas que pudieran ser implementadas en una segunda versión de MATfor.

Las nuevas características serían:

* Estructura completa y simple en una línea.

  Por ejemplo:

  ```
  WHILE (A.GT.B) DO X = Y + Z
  ```

* Instrucciones múltiples en una línea.

  Por ejemplo:

  ```
  CASE var OF ; 1,2,3 : A=3 ; 4,5,6 : A=6

  WHILE (X.LE.Y) DO ; BEGIN ; IF ...
  ```

* Traducción de los operadores relacionales y lógicos.

  Por ejemplo:

  ```
  >     ----------    .GT.

  <     ----------    .LT.
  .                     .
  .                     .
  .                     .

  AND                 .AND.
  ```

* Operaciones más complejas con vectores y matrices.

  Por ejemplo:

  ```
  E = ( > ( V * E ) )
  ```

  Multiplicará el vector "V" por el escalar "E" y obtendrá el máximo de ese vector resultante y lo dejará en el escalar "E".

```
V = ( V1 + ( ( M * V ) * E ) )
```

Multiplicará la matriz "M" por el vector "V", después el resultado, que es un vector lo multiplicará por el escalar "E", y este resultado lo sumará al vector "V1", quedando todo en "V".

- Implementación de más operaciones entre vectores y matrices.

  Por ejemplo:

  Transpuesta de una matriz.
  División de vectores.
  Determinante de una matriz.
  Etc.

MATfor es el inicio de una nueva filosofía en preprocesadores para la elaboración de programas. Involucra algunas de las características de los lenguajes de procedimientos y de cuarta generación y, sobre todo, evita el cambio a otro compilador.

La forma de ejecución y la creación de archivos internos están hechas con tarjetas de control, propias del sistema operativo EXEC-8, puesto que fue el equipo donde se desarrolló y probó el preprocesador. Si se pretendiera instalar a MATfor en otra máquina se tendría que encontrar el equivalente a las tarjetas de control en ese equipo y modificar las subrutinas en ensamblador, si fuera necesario, de acuerdo a la forma de operar de dicha computadora.

ANEXO A

" E R R O R E S "

** SUBROUTINE MAT **

```
ERROR!!... EN DECLARACION DE MATRICES
           MAT[1<.>N,1..M] ---> MAT[1..N,1..M]
ERROR!!... EN DECLARACION DE MATRICES PUNTO DE MAS
           MAT[1<...>N,1..M] ---> MAT[1..N,1..M]
ERROR!!... EN DECLARACION DE MATRICES
           MAT[1..N,1<.>M] ---> MAT[1..N,1..M]
ERROR!!... EN DECLARACION DE MATRICES PUNTO DE MAS
           MAT[1..N,1<...>M] ---> MAT[1..N,1..M]
ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "MAT"
ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "MAT"
ERROR!!... EN "MAT" TIPO DE VARIABLE NO CONSIDERADO
```

** SUBROUTINE VEC **

```
ERROR!!... EN DECLARACION DE VECTORES
           VEC[1<.>N] ---> VEC[1..N]
ERROR!!... EN DECLARACION DE VECTORES PUNTO DE MAS
           VEC[1<...>N] ---> VEC[1..N]
ERROR!!... MAXIMO 10 DIGITOS DE DIMENSION EN "VEC"
ERROR!!... NO SE ESPECIFICO DIMENSIONAMIENTO EN "VEC"
ERROR!!... EN "VEC" TIPO DE VARIABLE NO CONSIDERADO
```

** SUBROUTINE SEPARA **

```
ERROR!!... SINTAXIS EN DEC. DE "VAR" O "ARRAY"
ERROR!!... EN "VAR" O "ARRAY" VARIABLE MAYOR DE 6
ERROR!!... EN DECLARACION DE "VAR" O "ARRAY"
```

** SUBROUTINE FUNB **

```
ERROR!!... FUNCION NO DISPONIBLE
```

$$ SUBROUTINE COMPAT $$

```
ERROR!!... DIMENSIONES INCOMPATIBLES EN
           "V=V+V,V=V-V,V=V$V"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V$M"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=V$E"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=E$V"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M+M O M=M-M"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M$M"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "V=M$V"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=M$E"
ERROR!!... DIMENSIONES INCOMPATIBLES EN "M=E$M"
ERROR!!... DIMENSIONES INCOMPATIBLES
           MATRIZ CUADRADA PARA OBTENER SU INVERZA
ERROR!!... FUNCION NO CONSIDERADA EN SUB-COMPAT
```

$$ SUBROUTINE SUBSTA $$

```
ERROR!!... FUNCION NO CONSIDERADA
```

$$ SUBROUTINE PRIMI $$

```
ERROR!!... VARIABLE MAYOR DE 6 DIGITOS
           AL USAR FUNCION EXCLUSIVA DE $MATFOR$
ERROR!!... FUNCION DE $MATFOR$ DECLARADA INCOMPLETA
ERROR!!... FUNCION UNIVOCA DE $MATFOR$ MAYOR DE 6
           DIGITOS
ERROR!!... NO SE ENCONTRO SIGNO DE (=) EN FUNCION DE
           $MATFOR$ O > BLANCOS
ERROR!!... FUNCION DE $MATFOR$ DECLARADA INCOMPLETA
           CAMPO DE VARIABLE EN BLANCO
```

$$ SUBROUTINE QUESOY $$

```
ERROR!!... VARIABLE NO DECLARADA EN "VAR" O "ARRAY"
           Y UTILIZADA EN FUNCIONES DE $MATFOR$
ERROR!!... VARIABLES DE DIFERENTE TIPO EN FUNCIONES DE
           $MATFOR$ REALES/ENTERAS???
```

**** SUBROUTINE ARCHIV ****

```
ERROR!!... EN EL LLAMADO DEL PRE-PROCESADOR
           (FALTO EL PUNTO EN EL ARCHIVO DE ENTRADA)
```

**** SUBROUTINE INFORM ****

ERROR EN EL LLAMADO DEL PRE-PROCESADOR *MATFOR* DE SGC/RHR VER.01

**** SUBROUTINE PREPRO ****

```
ERROR!!... OPCION INCORRECTA (MATFOR,ABS O MATFOR,SYM)
ERROR!!... ARCHIVO DE ENTRADA NO CATALOGADO.
ERROR!!... ARCHIVO DE SALIDA NO CATALOGADO.
```

**** SUBROUTINE CASE ****

```
ERROR!!... MALA DECLARACION DEL "CASE" FALTO ESPACIO
ERROR!!... MALA DECLARACION DEL "CASE", *VARIABLE EN
           CUESTION??
ERROR!!... VARIABLE EN CUESTION DEL "CASE" MAYOR DE 6
           DIGITOS
ERROR!!... FALTO ESPACIO PARA EL "OF" EN EL "CASE"
ERROR!!... PARENTESIS INCOMPLETOS EN EL "CASE"
ERROR!!... MALA DECLARACION DEL "CASE" FALTO "OF"
ERROR!!... FALTO DECLARACION EJECUTABLE DESPUES DE LOS
           DOS PUNTOS
ERROR!!... FALTARON LOS DOS PUNTOS (:) O "END CASE"
ERROR!!... FALTO "END CASE"
ERROR!!... FALTO CERRAR EL BEGIN ABIERTO "END BEGIN"
```

$$ SUBROUTINE FOR $$

```
ERROR!!... MALA DECLARACION PARA EL "FOR"
ERROR!!... VARIABLE MAYOR DE 6 DIGITOS EN EL "FOR"
ERROR!!... DECLARACION DEL "FOR" INCOMPLETA FALTO "DO"
ERROR!!... FALTO ESPACIO PARA EL "DO" EN EL "FOR"
ERROR!!... NO SE ENCONTRO "ENDF" O "END FOR"
```

$$ SUBROUTINE REPEAT $$

```
ERROR!!... PARENTESIS INCOMPLETOS EN EL "REPEAT-UNTIL"
ERROR!!... FALTO CONDICION DEL "REPEAT-UNTIL"
ERROR!!... NO SE ENCONTRO "UNTIL" DEL "REPEAT"
```

$$ SUBROUTINE WHILE $$

```
ERROR!!...FALTO ESPACIO ENTRE EL WHILE Y LA CONDICION
          (  "WHILE (CONDI.)"  )
ERROR!!... FALTO CONDICION EN EL "WHILE"
ERROR!!... MALA DECLARACION DEL "WHILE" FALTO "DO"
ERROR!!... FALTO ESPACIO ENTRE "CONDICION" Y "DO"
ERROR!!... CONDICION DEL "WHILE" VA ENTRE PARENTESIS
ERROR!!... PARENTESIS INCOMPLETOS EN EL "WHILE"
ERROR!!... FALTO "ENDW" O "END WHILE"
```

$$ SUBROUTINE VAR $$

```
ERROR!!... FALTARON (:) EN DEC. "VAR"
ERROR!!... "VAR" O "ARRAY" SIN TIPO
           CHECAR SINTAXIS
```

**** ERRORES EN LA EJECUCION ****

```
@MATFOR,SYM KBB*SGCT12.,KBB*SGCSAL
  39  ILLEGAL CHARACTER  *
(FALTO EL PUNTO EN EL ARCHIVO DE SALIDA)

@MATFOR,SYM KBB*SGCT12.,KBB*SGCSAL.
MATFOR SGC/RHR VER.01 08/16/88-16:44:20
  ERROR!!... ARCHIVO KBB*SGCSAL(1). NO CATALOGADO.

@MATFOR,SYM KBB*SGCT122.
MATFOR SGC/RHR VER.01 08/16/88-16:44:44
  ERROR!!... ARCHIVO KBB*SGCT122(1). NO CATALOGADO.

@MATFOR,SYM
MATFOR SGC/RHR VER.01 08/16/88-16:44:49
ERROR EN EL LLAMADO DEL PRE-PROCESADOR *MATFOR* DE
SGC/RHR VER.01

@MATFOR,ABS SGCT12.,SGCSAL.
MATFOR SGC/RHR VER.01 08/16/88-16:45:10
  ERROR!!... ARCHIVO KBB*SGCSAL(1). NO CATALOGADO.

@MATFOR,ABS SGCT12
MATFOR SGC/RHR VER.01 08/16/88-16:45:15
  ERROR!!... EN EL LLAMADO DEL PRE-PROCESADOR
             (FALTO EL PUNTO EN EL ARCHIVO DE ENTRADA)

@MATFOR,ABS SGCT122.
MATFOR SGC/RHR VER.01 08/16/88-16:45:30
  ERROR!!... ARCHIVO KBB*SGCT122(1). NO CATALOGADO.

@MATFOR,ABS
MATFOR SGC/RHR VER.01 08/16/88-16:45:35
ERROR EN EL LLAMADO DEL PRE-PROCESADOR *MATFOR* DE
SGC/RHR VER.01

@MATFOR,OPT SGCT12.
MATFOR SGC/RHR VER.01 08/16/88-16:45:46
  ERROR!!... OPCION INCORRECTA (MATFOR,ABS O MATFOR,SYM)
```

ANEXO B

" B I B L I O T E C A S "

CAD/CAM.

ADAMS.

ADAMS es un paquete generalizado que da a los ingenieros y diseñadores la habilidad de el desarrollo rápido de modelos y el análisis eficiente de las características de los sistemas mecánicos. El sistema mecánico tratado, puede ser descrito generalmente como cuerpos rígidos interconectados con rigidez y elementos amortiguados, sujetos a una variedad de fuerzas externas y movimientos. Las paradas mecánicas,las características de los componentes no lineales y los grandes movimientos en estos sistemas de tres dimensiones da múltiples grados de libertad. Las capacidades analíticas incluyen KINEMATICS,STATICS, y DYNAMICS.

FILSYN.

Este programa es usado para el diseño, síntesis y análisis de todas las clases de filtros eléctricos. Los filtros que pueden ser manejados son: Filtros LC Passie (incluyendo filtros de cristal),filtros de microondas, filtros activos RC (incluyendo filtros de capacitor-switcheado) y filtros digitales (ambos de la variedad IIR y FIR).

El programa maneja todas las fases del procedimiento de diseño, desde la aproximación inicial a través de la síntesis, cálculo de los valores de los elementos, cualquier modificación y manipulación que sea necesaria a través del análisis, y también de la equalización de retardos.

FLOWTRAN. (2-172)

FlOWTRAN procesa material de estado estacionario (steady state) y de energia balanceada para procesos de ingeniería de procesos químicos basados en flúidos. Sus extensiones de desarrollo de procesos conceptuales a través del proceso de inicio, diseño y operación para procesar las modificaciones y la resolución de cuellos de botella.

PREDICTOR.

Es una herramienta de ingeniería de confiabilidad, y disponibilidad que le proporciona al ingeniero evaluador un método conveniente, rápido y preciso para la predicción de información electrónica y mecánica RAM. El paquete contiene un método comprensivo para alcanzar los requerimientos

MIL-HDBK-217. Este programa maneja confiabilidad de misión multifase, modos de misión de falla y efectos de análisis criticos, confiabilidad funcional y mucho mas.

SIGMA.

Es un software del sistema interactivo de dos y tres dimensiones, diseñado por el centro de cómputo de la oficina de diseño, para el diseño y manejo de formas complejas tridimensionales, basada en una representación polinomial de ondas y superficies derivadas de los métodos de Bezier. SIGMA es una herramienta flexible y eficiente para diseñar y trabajar con formas complejas.

BUSSINES MANAGEMENT.

ALADIN

Sistema de administración de bibliotecas. Permite el mantenimiento de bibliotecas y el control de los préstamos. los reportes estandars y específicos para nuevas adquisiciones, encabezados por temas, etc., revision de libros, búsquedas en línea y recuperación.

DYSMOD.

Es una herramienta avanzada para politicas de diseño en administración, el gobierno y la defensa. Este paquete automáticamente encadena modelos escritos por el paquete DYSMPOP con una característica de optimización sofisticada. Esto da al usuario acceso a técnicas potentes para explorar los efectos de parámetros en cambios estructurales, o la ejecución del sistema a ser analizado, o da una búsqueda completa y extensiva de las posibles mejoras del sistema.

PACK.

Este sistema es para análisis de series de tiempo, fue diseñado para alcanzar todas las necesidades de SOFTWARE correspondientes a box-jenkins o a la construcción de un modelo ARIMA. Desde la codificación del proceso Box-Jenkis, más y más analistas de series de tiempo están concordando en que esta técnica ofrece la mejor solución al problema de predicción. Esto es debido a que el método no asume que las observaciones son independientes, o que los residuos sean de ruido blanco (white noise).

SEISPACK.

Procesamiento de datos sísmicos incluyendo líneas de ondulación (wiggle lines), área de variables, cuadriculado, dos graficadores de atributos, documentado y vistas en pantalla.

UNIGRAPH.

Un sistema de graficación basado en un menú interactivo diseñado para usar las características de BIZPAK. Permite a las gráficas ser preparadas inmediatamente, desde los datos de entrada hasta una biblioteca de gráficas para actualización.

COMMUNICATIONS

CAPS (2-61)

Es una herramienta automatizada para el área de telecomunicaciones corporativas, proporciona análisis del uso del teléfono para un centro de costos, departamento y compañía. También proporciona análisis del uso de grupos de líneas y costos del equipo.

KERMIT (2-260)

Es un programa de transferencia de archivos de micro a macro, los archivos a transferirse se descomponen en paquetes con ckecksums y otra información de control para asegurar transmisiones completas y libres de error.

PREDICTOR BELL (2-403)

El método de confiabilidad de BELL se adhiere a la metología descrita en el BELLCORE (BELL COMMUNICATION RESEARCH), Publicación consultiva técnica IP-10425, Predicción de el procedimiento de confiabilidad para equipo electronico) de enero de 1984.

CONSTRUCCION

BESTWAY (2-42)

Dado un mapa de camino digitalizado, BESTWAYS calcula para cada elección interactiva de puntos de inicio y fin, el mejor camino para un vehiculo (distancia mínima o camino más rápido permitido para las calles de una sola via, o con restricciones de tráfico, tales como "no vuelta a la izquierda", límites de velocidad).

HCC-III

Calcula el diseño de las cargas de calentamiento y enfriamiento para construcciones de acuerdo a los métodos preescritos por ASHRAE.

HCCL-I

Es una versión del programa HCC-III mucho más fácil de usar. Esta versión es más económica, ya que caracteriza un método de entrada simplificado y un sistema único de exposiciones indexadas. los cálculos de carga son hechos hasta para doce meses o más, precisamante determinadas las condiciones de carga pico.

PENSB. (2-371)

Es un programa de análisis de tensión de tubería diseñado para manejar análisis dinámico y estático de los sistemas de tubería. El análisis estático incluye consideraciones termales, de peso, de viento, sísmicos, de fricción. El análisis dinámico incluye modo natural, espectro de respuestas, tiempo de historia de integración modal, integración directa y armónica.

SURVEY. (2-547)

Es un lenguaje orientado a problemas que utiliza la terminología familiar del topógrafo para ejecutar cálculos de topografía y análisis de subdivisiones. Una biblioteca extensiva de comandos permite al usuario solucionar cualquier tipo de cálculo geométrico o problema complejo.

EDUCATION

A88D. (2-33)

Es un macro ensamblador para el ensamblador IBM/370. Está hecho para el entrenamiento de estudiantes. Emula al IBM/370 en que el usuario puede obtener resultados, dumps de registros y memoria con un módulo del programa ensamblador como si estuviera trabajando en la IBM/370 .

DYBMAP.

Es un lenguaje para descripción, representación y simulación dinámica de cualquier problema socioeconómico o de administración, y es capaz de representar efectos contínuos, discretos y estocásticos. El paquete proporciona análisis de chequeo de error, documentación del módelo, análisis dimensional, y salida flexible, incluyendo gráficas elegantes Calcomp.

FORTIM. (2-175)

Determina las porciones de un programa FORTRAN V que son consumidoras de tiempo de CPU. FORTIM es muy útil para optimizar los programas de producción.

INFOFETCH. (2-246)

Es un software de cuarta generación, integrado y diseñado especialmente por CYBER. Es un generador de aplicaciones utilizado ampliamente, un generador de código y un sistema administrador de bases de datos relacional, diseñado para usuarios desde técnicos hasta los más capacitados programadores.

SHAZAM. (2-480)

Es un amplio paquete econométrico. Incluye análisis de regresión, estimación de ecuaciones simultáneas, componentes principales, y una extensa capacidad de manipulación de datos.

RATFOR. (1-101)

Es un preprocesador para fortran racional. Aunque FORTRAN no es siempre un lenguaje agradable de usar, tiene las ventajas de universalidad y eficiencia. RATFOR intenta ocultar las principales deficiencias mientras que retiene sus cualidades deseables y proporciona un buen control de flujo de operaciones.

ELECTRONIC CAE/CAD

ASPEC. (1-10)

Es un programa de simulación de circuitos electrónicos de propósito general capaz de ejecutar los siguientes tipos de análisis:

- CD No lineales
- Transitorios no lineales
- Lineales de señales pequeñas AC

Además, cada tipo de análisis puede ser repetido automáticamente a varias temperaturas determinadas por el usuario o condiciones de casos adversos determinados por el usuario.

MIDAS. (1-71)

Es un conjunto integrado de aplicaciones de diseño automático, usado en ingeniería y manufactura para ayudar entre otras cosas a:

- Diseño a nivel de sistema funcional.
- Diseño lógico.
- Verificación del diseño lógico AND/OR utilizando simulación de computadora.
- Partición del diseño lógico en IC,módulos etc.

OPTIMOS. (1-83)

Proporciona al ingeniero de diseño la capacidad de modelar dispositivos MOS, extracción de parámetros y verificación de dispositivos para uso en el programa de simulación de circuitos ASPEC.

PREDICTOR. (2-402)

Es una herramienta de ingeniería confiable, disponible y mantenible (RAM) que proporciona al ing. evaluador un método rápido, conveniente y preciso para la predicción de la información electrónica y mecánica RAM.

SUPERFILSYN. (1-117)

Es una aplicación de síntesis de filtros para el uso de los ingenieros electrónicos que diseñen redes de filtros activos y pasivos. La entrada es la característica deseada del filtro. La salida es el conjunto de parámetros de la red de los filtros.

ENGINEERING.

ADORE. (2-12)

Proporciona una simulación de tiempo real de la ejecución dinámica de las cargas de rodaje.

ANSALT. (2-24)

Ejecuta avance estructural y análisis de transferencia de calor especialmente en el área de procesos termomecánicos relacionados al almacenamiento o liberación de formaciones salinas.

BEASY. (2-40)

Es un programa de computadora de propósito general que soluciona una amplia variedad de problemas de análisis de ingeniería utilizando el método de elementos límites.

BRB2. (2-209)

Un sistema de optimización de propósito general, totalmente en FORTRAN, portable. Puede ser usado como un sistema aislado o llamado como una subrutina

SINDA. (2-492)

Es un paquete de software diseñado para analizar sistemas térmicos de fluidos representados en forma de parámetros reunidos. Puede manejar conducción, convección y radiación y tiene la habilidad para modelar fuentes de calor o sumergidas, variando las condiciones límites y las propiedades no lineales de los materiales.

GRAPHICS/PLOTTING

ASPEX. (2-32)

Un programa para desplegar datos en un cuadriculado como una superficie vista en perspectiva oblícua. ASPEX produce vistas perspectivas de superficies de tres dimensiones en modo interactivo o en modo batch sobre un TRC o con una pluma de graficador.

MAPEX. (1-67)

Es un software de aplicación para la creación de mapas de base para ayudar a los geólogos y geofisicos en la interpretación y actividades de exploración.

PLOT 10 GKS . (1-96)

Es una biblioteca de subrutinas FORTRAN que sigue los estandares GKS (nivel 2B). Este paquete provee al desarrollador de aplicaciones con una amplia variedad de primitivas de salida y de creación de gráficas, segmentos y textos en dos dimensiones.

SIMPLOT. (2-489)

Es un paquete en FORTRAN 77 de subrutinas y programas que permite dibujar líneas (sencillas y amplias), líneas curvas, texto, líneas débiles, símbolos autodefinidos, etc.

UNIGRAPH. (2 -629)

Un sistema de graficación conducido por un menú interactivo para utilizar las caracteristicas de BIZPAK (BIZMAP opcional). Permite que las gráficas se preparen inmediatamente de los datos de entrada o de una biblioteca de gráficas.

ANEXO C

" F O R T R A N 8 X "

Major Emphasis on Modernization

* Free source form

* Statement enhancements

* New statements and features
  - New statements
  - Array operations
  - User-defined types
  - Modules
  - Procedures
  - Precision control

* Standardized old friends : NAMELIST

* Mechanism to delete unused features

MISCELLANEOUS ENHANCEMENTS

* 31 -character names, underscore

* Symbolic relational operators

.EQ.	==	.LT.	<	.GT.	>
.NE.	<>	.LE.	<=	.GE.	>=

* String delimiter:apostrophe(') or quotation mark (")

FREE SOURCE FORM

* Lines up to 132 characters

* Comment indicator: !
```
x = 3.14159 * r**2 ! Compute area
```

* Continuation indicator: &
```
100 FORMAT(1x,'The lines begins here &
            & and ends here.')
```

* Statements separator: ;
```
n = 1 ; a = 3.2
```

TYPE STATEMENTS

```
* FORTRAN 77

      INTEGER                 a(10), b(10), c(20)
      INTEGER                 x,y
      DATA                    x/1/,y/2/
      CHARACTER*3             c1,c2*6

* FORTRAN 8X

      INTEGER,ARRAY(10):: a,b,c(20)
      INTEGER DATA      :: x=1,Y=2
      CHARACTER(LEN=3)     c1,c2(len=6)
```

OTHER SPECIFICATION STATEMENTS

```
* FORTRAN 77

      Real                    one,p
      Parameter               (one = 1.0,p = 4.1/3.0)

* FORTRAN 8X

      Real, Parameter :: one = 1.0, p = 4.1/3.0

      Integer, array(3),parameter ::order = (1,2,3)
      Real(precision = 10, exponent-range = 50) t1
```

DATA STATEMENT

```
* FORTRAN 77 (List - oriented)

      Data k/2/
      Data ((b(m,n),m= 1,5),n=1,5),j /20*4.1,6*5.1/

* FORTRAN 8X (Object-oriented)

      Data (k=2)
      Data (((b(m,n),m=1,5),n=1,5)= ( 20(4.1),5(5.1)),&
           j= 5.1)
```

IMPLICIT STATEMENT

```
# IMPLICIT NONE
```

DO CONSTRUCT

```
# (name:) DO -statement

          END DO (NAME)

# DO, n=2,100,2
# DO (k) TIMES
# DO
```

EXIT,CYCLE (for DO)

```
# Exit (name)
# CYCLE (name)

# outer: DO (k) TIMES
# inner:  DO

              :::::
              IF (exit-condition) EXIT
              IF (cycle- condition)CYCLE outer
          END DO inner
          : : : :
        END DO outer
```

DO LOOPS

```
# Poor man's DO WHILE
    DO
         IF (end-condition) EXIT
    : : : :
    END DO

# Poor man's DO UNTIL
    DO
    : : : :
         IF ()END-CONDITION) EXIT
    END DO
```

IF CONSTRUCT

```
(name:) IF (condition) THEN
     : : : :
          ELSE IF (condition) THEN (name)
               : : : :
               ELSE (name)
                    : : : : :
          END IF (name)
```

SELECT CASE CONSTRUCT

```
# (name:) SELECT CASE (case-expression)
          CASE case-selector  (name)
               block
               : : : : :
          END SELECT (name)
```

SELECT CASE CONSTRUCT

```
 (name:) SELECT CASE (case-expression)
          CASE case-selector    (name)
               block
               : : : :
          END SELECT   (name)
```

case-expression	:	integer,character,logical
case-selector	:	(case-value-list) or DEFAULT
case-value	:	integer,character,logical
	:	constant expression
	:	value,value:, :value,value

SELECT CASE CONSTRUCT

```
SELECT CASE (i)
       CASE( :-1)                    ; Negative values

       CASE (0)                      ; Just zero

       CASE(1:10)                    ; Values 1 to 10

       CASE(20,30,40)                ; Individual values

       CASE(51:55,58)                ; Combination

       CASE DEFAULT                  ; Any other values

     END SELECT
```

ARRAY OPERATIONS

* Whole, partial and subset array operations
* Dynamically allocatable arrays (ALLOCATE,DEALLOCATE)
* Conditional array assignment (WHERE)
* Assumed-shape arrays
* Array intrinsics
* Array constructors

WHOLE ARRAY OPERATION

* FORTRAN 77

```
     Real    a(40),b(40)
     do 10, i=1,40
  10 .... a(i) = 0.0
  20 .... a(i) = a(i) + 3.0*sin(b(i))
```

* FORTRAN 8X

```
     Real,array(40) ::a,b
     a = 0.0
     a=a +3.0*sin(b)
```

PARTIAL ARRAY OPERATIONS (SECTIONS)

```
* FORTRAN 77
      Real a(40),b(40)
      Do 10,i=1,40
   10 ...   a(i)= 0.0
      Do 20,i=1,n,nstep
   20       a(i)=a(i) + sin(b(i))

* FORTRAN 8X

      Real a(40), b(40)
      a = 0.0
      a(1:n:nstep)+SIN (b(1:n:nstep))
```

ARRAY SUBSETS (IDENTIFY)

```
INTEGER,ARRAY(4,4)        :: a
REAL,ARRAY(:),ALIAS       :: diag
IDENTIFY ( diag(i)  = a(i,i), i=1:4 )

      1     0     0     0
      0     1     0     0
      0     0     1     0
      0     0     0     1
```

ARRAY SUBSETS (IDENTIFY)

```
INTEGER,ARRAY(4,4)        :: a
REAL,ARRAY(:,:), ALIAS    :: llq
IDENTIFY ( llq(i,j)   = a(i+2,j), i=1:2, j=1:2 )

      0     0     0     0
      0     0     0     0
      1     1     0     0
      1     1     0     0
```

ARRAY SUBSETS (SET RANGE)

```
REAL,ARRAY(40),RANGE  /group/      ::a,b
REAL,ARRAY(40),RANGE               ::c

      SET RANGE(n   : n+10)        /group/
      SET RANGE(n, : n+10)         c

      a = 0.0
      b = a*SIN(c)
```

DYNAMICALLY ALLOCATABLE ARRAYS

```
REAL,ARRAY(:,:),ALLOCATABLE       ::a,b
ALLOCATABLE( a(i+2,0:j), b(5,5) )
: : : :
DELLOCATABLE( a )
: : : :
DELLOCATABLE( b )
```

CONDITIONAL ARRAY ASSIGNMENT

```
WHERE Statement
* FORTRAN 77
      Real   a(10,10), b(10,10)
      Do 10,j=1,10
            Do 10,i=1,10
                  If (b(i,j) .NE. 0.0)  a(i,j)=a(i,j)/b(i,j)
10 Continue

* FORTRAN 8X
      Real   a(10,10), b(10,10)
      Where (b .NE. 0.0) a=a/b
```

CONDITIONAL ARRAY ASSIGNMENT

```
WHERE Construct


REAL a(10,10), b(10,10)
WHERE (b .NE. 0.0)
      a=a/b
ELSEWHERE
      a=0.0
END WHERE
```

ASSUMED-SHAPE ARRAYS

```
ARRAY(:,:),ALLOCATABLE      ::a
ARRAY(50,50)                ::b
: : : : :
ALLOCATABLE( a(m,n) )
CALL sub( a )
CALL sub( b(2:50:2,:) )

SUBROUTINE sub( arg )
     ARRAY (:,:) :: arg
```

ARRAY INTRINSICS

```
* Categories:Elemental,Inquiry,Transformational
* Elemental
- ABS,CMPLX,MIN,MAX,SIN,COS,....

* Inquiry
- ALLOCATED,DLBOUND,ELBOUND,DSIZE,ESIZE,....

* Multiplication and reduction
- Dotproduct,Matmul,Product,Sum,Any,All,......

* Construction and manipulation
- Merge,Spread,Pack,Unpack,Transpose,.........
```

ARRAY INTRINSICS

```
* Fourier sum
     F = Σ ai cos xi

     F = SUM(a*COS(x) )

* Conditional Fourier Sum
     CF = Σ ai cos xi
          |ai| < 0.01
     CF = SUM (a*COS(x),MASK = ABS(a) < 0.01)
```

ARRAY CONSTRUCTORS

```
(value-list) or (/value-list/)
INTEGER,PARAMETER  :: order(3) = (6,10,15)
INTEGER            :: save(3)

IF (condition) THEN
     save = order
ELSE
     save = (/3,4,5/)
END IF
```

ARRAY CONSTRUCTORS

```
INTEGER,ARRAY(:,:),ALLOCATABLE    :: a
ALLOCATABLE( a(m,n) )
a = RESHAPE( (m,n),      &
               (m:1:-1,0,v1+v2, k+1 (1,2,3) ))
```

PROGRAMMER-DEFINED DATA TYPES

```
* Called "derived types"
* A means of producing structures
* Assignment
* Operators: defined operators
             overloaded intrinsic operators
```

DEFINITION AND DECLARATION

```
TYPE var-string (max-len)
     INTEGER                          current-lenght
     CHARACTER(LEN = max-len)         string-value
     TYPE (var-string(20))   ::       s1,s2
     s1 = s2
```

ASSIGNMENT OF VALUES

```
* Direct assignment to individual fields
* Structure constructor
* Assignment subroutine
```

DIRECT ASSIGNMENT

```
* Direct assignment to individual fields
* Structure constructor
* Assignment subroutine

          s1%current-lenght = LEN("Fortran 8x   ")
          s1%string-value = 'Fortran 8x    '
```

STRUCTURE CONSTRUCTOR

```
* Direct assignment to individual fields
* Structure constructor
* Assignment subroutine

     CHARACTER(LEN=12)      char-val
     char-val = 'Fortran 8x   '
     s1 = var-string (20)   (LEN(char-val),char-val)
```

ASSIGNMENT SUBROUTINE

```
* Direct assignment to individual fields
* Structure constructor
* Assignment subroutine
     Subroutine assign-to-string(string,chars) Assignment
     Type(var-string(*))       string
     Character(*)              chars
     string%current-length= LEN-TRIM(CHARS)
     string%string-value(1:string%current-length)= chars
     END SUBROUTINE assign-to-string
     s2 = "Fortran 8x   "
```

DEFINED OPERATORS

```
* Form
     - Same as relational operators
     - Up to 31 letters
     - Example: .COMBINE.

* Procedence
     - Unary: Highhest of all operators
     - Binary: Lowest of all operators

* Operation is via an operator function
```

DEFINED OPERATORS

```
Type(var-string(20))  ::s1,s2
If (s1 .SEQ. s2) Print *,"they're equal"


Logical Function str-eq(str1,str2)Operator  (.SEQ.)
     Type(var-string(*))  ::str1,str2
     str-eq = str1%string-value(1:str1%current-length) .EQ. &
              str2%string-value(1:str2%current-length)
End
```

OVERLOADED INTRINSIC OPERATORS

```
* Can only overload when operands are not defined for
  operator
* Retains precedence of intrinsic operator

     Type(var-string(20))  :: s1,s2
     If (s1 .EQ. s2)print *,"they're equal"

     LOGICAL FUNCTION str-eq(str1,str2)Operator(.EQ.)
          Type(var-string(*))  :: str1,str2
          str-eq = str1%string-value(1:str1%current-length)
          .EQ. &
          str2%string-value(1:str2%current-length)
End Function
```

COMBINE WITH INTRINSIC TYPES

```
Type(var-string(20))                  :: s
Character(len=16)                     :: c
If (s = = c)print *,"they're equal"

LOGICAL FUNCTION str-char-eq(string,chars)Operator(= =)
     Type(var-string(*)                :: string
     Character(*)                      :: chars
     str-char-eq =           &
          string%string-value(1:string%current-length) .EQ.
          chars
End Function str-char-eq
```

GLOBAL DEFINITIONS/PROCEDURES

```
* INCLUDE not in standard
* Module concept
     - Not textual subtitution
     - May contain data declarations and/or procedures
     - Data/procedures may be "private"
     - Module may be separately "compiled"
     - Introduces separate compilation
     - Replacement for block data,common blocks,ENTRY
     - statements
```

MODULES

```
MODULE aaa
     REAL,PRIVATE              :: x,y
     CHARACTER                 :: c1(6),c2(12)
     CONTAINS
     SUBROUTINE check(arg1,arg2)

     : : : : : : : :
     END SUBROUTINE
END MODULE
```

MODULES

```
* Accessed via USE statement
* May select specific data from module
* May rename data accessed from a module

     Module abc                        Module def
          Real           :: x,y             Contains
          Real,private:: z                  Subroutine x
     End Module abc                         : : : : :
                                            End Subroutine x
     Use  abc,only : x                      End Module def
     Use  def, x => xx
```

PROCEDURE INTERFACE DESCRIPTION

```
* Safety of programming
* Interface block describes arguments
* Not required for 77-style arguments
* Needed for noncontiguos array arguments,optional
- arguments,keyword arguments,assignment subroutines,operator
- functions,...
* Only needed for external procedures
```

INTERFACE BLOCK

```
Interface
     Subroutine expand(array,i,j,flag)
     Use expand-module
     Implicit Complex(a)
     Array(i,j)                  ::array
     Integer                     ::i,j
     Integer Optional            ::flag
End Interface

Complex           calc(5,6)
: : : : :
Call  expand(calc(2:4:2,:)m,n)
```

INTERNAL PROCEDURES

```
* Essentially same rules as FTN/UFTN
- (Scope of names,IMPLICIT rules,single-level nesting)

          Program Eddy
          i=0
          Call inc(i)
          Print*,i
          Contains
          Subroutine inc(iarg)
               iarg = iarg +1
          End subroutine inc
     End Program Eddy
```

MISCELLANEOUS PROCEDURE FEATURES

```
* Recursion
      RECURSIVE INTEGER FUNCTION factorial(arg)
* User elemental functions
* Argument keywords
      Call sub(ARRAY =a,LBOUND=1,UBOUND=M)
      Call sub(LBOUND=1,UBOUND=M,ARRAY=a)
* Optional dummy arguments  (OPTIONAL)
* Usage of dummy argument
      INTENT(IN),INTENT(OUT),INTENT(INOUT)
```

PRECISION CONTROL

```
* Real(precision,exponent-range)
 precision          ! minimum number of decimal digits
 exponent-range     ! minumum decimal exponent range
                     (+ - power of 10)
* Example ! REAL(8,35)
* May be (restricted) expressions or *
* Replaces single/double precision concept
```

PRECISION CONTROL

```
* New data type
* Never the same as default real
      Real(10,100) actual-arg
      Call sub(actual-arg)

      Subroutine sub(dummy-arg)
      Real dummy-arg          ;INVALID
* Can not be used in COMMON/EQUIVALENCE
```

PRECISION CONTROL

```
* Real(*,*)may only be dummy argument
* All Real(*,*)dummy args must have same precision
      Real(8,35) arg835-1,arg835-2
      Real(16,70) arg1670-1,arg1670-2
      Call sub(arg835-1,arg835-2)
      Call sub(arg1670-1,arg1670-2)
      Subroutine sub(dum1,dum2)
            Real(*,*) dum1,dum2
```

DECREMENTAL FEATURES

- All of FORTRAN 77 is included in 8x
- Part of languaje evolution/modernization safety of programming
- News reports are WRONG
- Appendix A!

  "Marking a feature as obsolescent or deprecated does not imply its removal from subsequent standards; notification is given that these features may be removed in subsequent revisions."

DECREMENTAL FEATURES

- 3-step process

  DEPRECATED -> OBSOLESCENT -> DELETED

DECREMENTAL FEATURES

- 3-step process

  Deprecated -> Obsolescent -> Deleted

  "... those features of FORTRAN 77 that are redundant and considered largely unused."

DECREMENTAL FEATURES

- 3-step process

  Deprecated -> Obsolescent -> Deleted

  "... those features of FORTRAN 77 that are redundant and for which better methods are available in FORTRAN 77."

DECREMENTAL FEATURES

Obsolescent feature	USE
Arithmetic IF	If contruct/stmt
Real/double precision DO control variable	Integer
Shared DO termination	CONTINUE for each DO
Termination on other than CONTINUE	CONTINUE for each DO
Branch to END IF from outside IF construct	Branch to stmt following END IF
Pause	Dummy read
Format label assigned to integer	Nothing (don't do it)

DECREMENTAL FEATURES

* 3-step process
Deprecated -> Obsolescent -> Deleted

"... expected to become obsolescent as the new features of this revision of the Fortran language become widely used."

* Storage association
* Redundant functionality

DECREMENTAL FEATURES

* Storage association
- Assumed-size dummy arrays
- Passing array element or substring to dummy array
- BLOCK DATA program unit
- COMMON and EQUIVALENCE statements
- ENTRY statements

DECREMENTAL FEATURES

DEPRECATED FEATURE	USE
Fixed source form	Free source form
Statement function	Internal procedure
Computed GO TO	SELECT CASE
List-oriented DATA	Object-oriented DATA
DIMENSION Statement	Type declaration (ARRAY)
DOUBLE PRECISION	Precision control (p,e)
*char-length	LEN = char-length
Specific intrinsic function name	Generic name

SUMMARY

* Emphasis on modernization
  - Array operations
  - Improved numerical computation facilities
  - User-defined data types
  - Modular data/procedure facilities
  - Language evolution concept
  - Free source form,more control constructs,recursion,
  - dynamically allocated arrays
* Near future:process comments
* Future: pointers?,bit?,multibyte characters?,varying length strings?,INCLUDE?

FORTRAN 8X STANDARDIZATION

Reasons for UNISYS NO VOTES

* Too much of an increment over FORTRAN 77
* Too much language design,
- Too little standardizing common practice
* Major new features are untested
* Larger language = larger compiler + larger RTS
* Degradation in compile-and run-time performance
* Deletion of storage association unrealistic

B I B L I O G R A F I A

1.- CREGS, Paul; DIRKSEN, Paul y GRAHAM, J. Wesley, "FORTRAN IV WITH WATFOR AND WATFIV" , 1970, Prentice-Hall, Printed in U.S.A., P. 5-9 .

2.- CONTROL DATA, "APPLICATIONS DIRECTORY", Fifth edition, April, 1986, Cyber 180 computer system.

3.- GORDON B.,Davis y HOFFMANN,Thomas R., "FORTRAN A STRUCTURED DISCIPLINED STYLE", Mc Graw-Hill, Book company, P. 285-287.

4.- GROGONO, Peter, "PROGRAMACION EN PASCAL", 1984, Fondo Educativo Interamericano, México.

5.- HOLT,R.C.; GRAHAM,G.S.; LAZOWSKA,E.D. y SCOTT,M.A., "STRUCTURED CONCURRENT PROGRAMMING WITH OPERATING SYSTEMS APPLICATIONS",1978, Adison-Wesley Publishing Company, P. 11.

6.- HATZAN , Harry Jr., "FORTR AN 77", Comput er Science Series VHR, Van Nostra nd Reinhold Compan y, 1978, P. V-VII,1-3.

7.- MOCK, Theodore J. ; VASARHELYI, Miklos A., "PROGRAMACION APL PARA ADMINISTRACION", 1977, Limusa, México.

8.- PLAUGER, Kernighan, "SOFTWARE TOOLS", 1976, Adison-Wesley Publishing Company, P. 285-287.

9.- ROLISON, Lawrence R., "8X: FUTURE OF FORTRAN", Aparecio en USE SPRING CONFERENCE, Volume I, Mayo, 1988, Baltimore Maryland, 1988 Unisys Corporation, Printed in U.S.A., P. 119-183.

10.- UNIVAC, "FUNDAMENTALS OF FORTRAN", Programer reference, UPE-7536 rev.1, Sperry Univac, 1974, P. 1-2.

11.- UNIVAC, "FORTRAN (ASCII) REFERENCE", Series 1100, Tomos I y II, UP-8244.3, Sperry Univac, 1985.

12.- UNIVAC, "META-ASSEMBLER (MASM)", Programer reference, Series 1100, UP-8453 rev.1, Sperry Univac, 1977.

13.- WEGNER,Peter, "PROGRAMMING LANGUAGES-THE FIRST 25 YEARS", IEEE Transaction on computer, Volume c-25 Num. 12, December, 1976, P. 1207-1208.