UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ciencias

"DESARROLLO DE UN LENGUAJE PARA EL ANALISIS NUMERICO Y SU TRADUCTOR"

T E S I S

QUE PARA OBTENER EL TITULO DE:

M A T E M A T I C O

P R E S E N T A :

MA. DEL ROSARIO SOLIS RODRIGUEZ

México, D. F. 1985

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

I. INTRODUCCION

1.1. Conveniencias de un macro-procesador para el análisis numérico

La finalidad de realizar un trabajo que sea una herramienta útil para el Análisis Numérico, es la de darle al usuario un método sencillo de resolver sus problemas relacionados con Métodos Numéricos.

Usualmente el usuario cuando tiene un problema en el área de Análisis Numérico, se escribe un programa en Fortran que le resuelva éste y eso lo tiene que hacer con cada problema que se le presenta. Se observa que hay procedimientos que se repiten en varios programas por lo que se crearon paquetes de programas, cuya única finalidad es la de ayudarle al usuario a resolver los problemas de una forma sencilla.

El usuario necesita conocer el lenguaje Fortran, para crear el programa que llama al paquete de programas que le resuelve el problema. A los programas que integran este paquete que por lo general vienen escritos en Fortran los llamaremos subrutinas o funciones.

Una subrutina es un procedimiento que se realiza sin retornarle al que lo llama valor alguno, mientras que la función, realiza algún procedimiento y retorna al programa principal un valor.

Otra de las cosas que se observan es que existen bloques de instrucciones que se repiten varias veces en el mismo programa, lo cual es un trabajo tedioso para los programadores. Dichos bloques pueden tener ligeras diferencias entre sí, estas diferencias pueden ser desde algún --

número en alguna instrucción, hasta un conjunto de instrucciones.

La similitud entre los bloques de instrucciones hace conveniente declarar dichos bloques de una sola vez, como un patrón con argumentos que varían en las diferentes instancias del uso de dicho patrón.

A las instrucciones que causan la generación de patrones de instrucciones se les conoce como Macro-instrucciones o simplemente Macros.

En términos muy generales se podría decir que un Macro da la facilidad de reemplazar una secuencia de símbolos por otra.

La estructura para la definición de una Macro-instrucción es:

```
MACRO NOMBRE (LISTA DE PARAMETROS)
CUERPO
ENDM
```

En donde MACRO y ENDM son normalmente palabras reservadas que determinan el inicio y el fin de una Macro-instrucción respectivamente. A esto también se le da el nombre de Macro-definición.

La cadena NOMBRE es un identificador asignado al Macro de tal manera que el ususario pueda definir varios patrones identificándolos con nombres diferentes.

El CUERPO es el bloque de instrucciones que determina el patrón en sí.

La LISTA DE PARAMETROS son los llamados parámetros formales que serán substituidos dentro del cuerpo del Macro por los parámetros actuales que serán dados por el usuario a la hora de hacer uso del Macro. Al hecho de hacer uso del Macro se le llama Macro-llamada y origina una Macro expansión.

La forma común de hacer un llamado a un Macro es:

NOMBRE (LISTA DE PARAMETROS ACTUALES)

Es decir se da el nombre del Macro (NOMBRE) al cual se quiere uno referir, así como la lista de parámetros actuales que serán los que sustituyan en el cuerpo del Macro a los parámetros formales.

Algunos Macros a los cuales nos estamos refiriendo, son usados en varias instancias, será necesario tenerlos en un archivo definitivo para que cada vez que se realice una Macro-llamada, revise ese archivo y lo pueda expander. A este archivo definitivo que contendrá los Macros así como los paquetes de programas (subrutinas o funciones) es a lo que en adelante llamaremos Bibliotecas.

De lo anterior, se ve que con las facilidades de Macros y los paquetes de programas (subrutinas o funciones), el usuario cuenta con dos herramientas poderosas a las cuales llamaremos Bibliotecas de Análisis Numérico y Macros.

Al sistema que manejara las Bibliotecas, de acuerdo a lo indicado por el usuario, dependiendo del problema a resolver se llamará Macro-procesador. A este Macro-procesador lo llamaremos de Análisis Numérico, por el tipo de Bibliotecas que maneja.

En la Figura 1.1. se muestra un esquema general de lo que son los Macro-procesadores, en este caso particular, el Macro-procesador de Análisis Numérico, el cual tienen como entradas 2 Bibliotecas y el programa del usuario. Tiene como salida un programa que contiene únicamente instrucciones de Fortran al cual llamaremos lenguaje Base.

Como podemos ver éste es un paso anterior al compilador de Fortran, es por eso que también recibe el nombre de preprocesador.

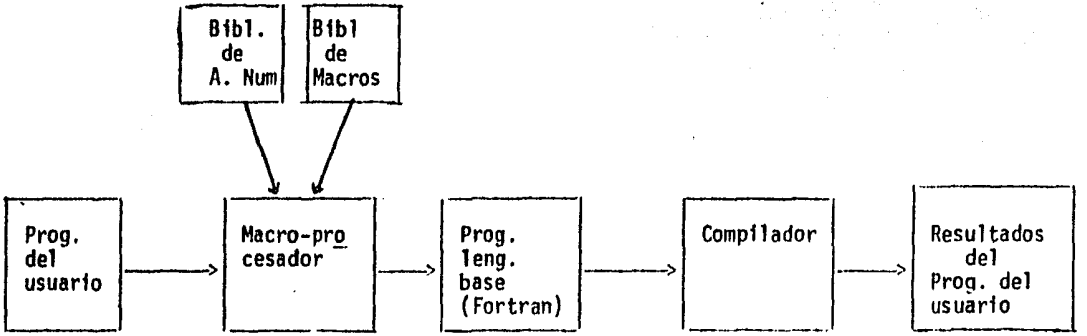

Bibl. de A. Num
Bibl de Macros
Prog. del usuario
Macro-pro cesador
Prog. leng. base (Fortran)
Compilador
Resultados del Prog. del usuario

Fig. 1.1

1.2. Ejemplos específicos del uso de macroprocesadores de Análisis Numérico

Para aclarar más la conveniencia de las aplicaciones del Macroprocesador de Análisis Número (MPAN) se verán algunos ejemplos:

Supóngase que un usuario desea sumar 2 Matrices de 10 x 10 cada una este usuario tiene dos caminos con los cuales llegará al mismo resultado;

Uno sería escribir un programa de FORTRAN

```
      DIMENSION A (10, 10)
      DIMENSION B (10, 10)
      DIMENSION C (10, 10)
      DO 100 X = 1, 10
      DO 100 Y = 1, 10
      READ (5, 500) A (X, Y)
100   CONTINUE
      DO 200 X = 1, 10
      DO 200 Y = 1, 10
      READ (5, 500) B (X, Y)
200   CONTINUE
      DO 300 X = 1, 10
      DO 300 Y = 1, 10
      C (X, Y) = A (X, Y) + B (X, Y)
300   CONTINUE
      DO 400 X = 1, 10
      DO 400 Y = 1, 10
      WRITE (5, 600) C (X, Y)
400   CONTINUE
500   FORMAT (16)
600   FORMAT (3X, I6)
```

Otro camino sería revisar en la Biblioteca del Macro-procesador del Análisis Número, con qué herramientas contamos para la resolución de nuestro problema.

Supongamos que en la lista de Macros tenemos:

%DEFMAT (A, I, J) define la dimensión de una Matriz A de I x J

%LEEMAT (A, I, J) lee una Matriz A de I x J

%ESCMAT (A, I, J) escribe la Matriz A de I x J

Y en las subrutinas encontramos:

%SUMA (A, B, C) suma la Matríz A a la de B y la asignamos a la Matríz C.

Entonces el programa quedaría así:

```
%DEFMAT (A, 10, 10)
%DEFMAT (B, 10, 10)
%DEFMAT (C, 10, 10)
%LEEMAT (A, 10, 10)
%LEEMAT (B, 10, 10)
%SUMA (A, B, C)
%ESCMAT (C, 10, 10)
```

Hay que hacer notar que el programa que quedó por el segundo método es mucho más sencillo y más entendible.

El Macro-procesador tiene como entrada este programa en un lenguaje que llamaremos fuente, por estar formado por instrucciones que generan Macro-llamadas o Macro-instrucciones. El cual al analizar cada una de las Macros o subrutinas llamadas, genera una expansión, quedando el programa de salida en lenguaje base de la siguiente manera:

```
      DIMENSION A(10, 10)          .....   %DEFMAT (A, 10, 10)
      DIMENSION B(10, 10)          .....   %DEFMAT (B, 10, 10)
      DIMENSION C(10, 10)          .....   %DEFMAT (C, 10, 10)
      DO 100 X = 1, 10             :
      DO 100 Y = 1, 10             :
      READ (5, 200) A(X, Y)        : .....   %LEEMAT (A, 10, 10)
100   CONTINUE                     :
200   FORMAT (18)                  :
      DO 300 X = 1, 10             :
      DO 300 Y = 1, 10             :
      READ (5, 400) B(X, Y)        : .....   %LEEMAT (B, 10, 10)
300   CONTINUE                     :
400   FORMAT (I8)                  :
      DO 500 X = 1, 10             :
      DO 500 Y = 1, 10             :
      C(X, Y) = A(X, Y) + B(X, Y): .....   %SUMA (A, B, C)
500   CONTINUE                     :
      DO 600 X = 1, 10             :
      DO 600 Y = 1, 10             :
      WRITE (5, 700) C(X, Y)       : .....   %ESCMAT (C, 10, 10)
600   CONTINUE                     :
700   FORMAT (3X, I6)              :
```

Este programa si lo comparamos con el programa que haría el usuario, es casi igual, tiene sus pequeñas diferencias, pero lógicamente realizan los mismo, es decir, al crear el programa objeto de los dos programas y correrlos, nos darían los mismos resultados.

Este ejemplo nos da una idea de lo importante que es contar con herramientas para permitir la generación de programas Fortran.

II. COMPARACION DE MACRO-PROCESADORES RATFOR Y MPAN

2.1. Introducción

Los Macro-procesadores tienen en la actualidad los siguientes usos:

a) El traslado software de una computadora a otra. El traslado de software es en general muy utilizado en cualquier tipo de programación, ya sea en paquete o programas de Sistemas Operativos.

Por ejemplo, si en la máquina A se diseñó un lenguaje LEN1 utilizando Macros y se desea pasarlo a la máquina B entonces se utiliza el Macro-expansor con los Macros respectivos. Más aún supóngase que el Macro-expansor se desarrolle en A, esto desembocaría en un traslado total, aún del Macro-expansor.

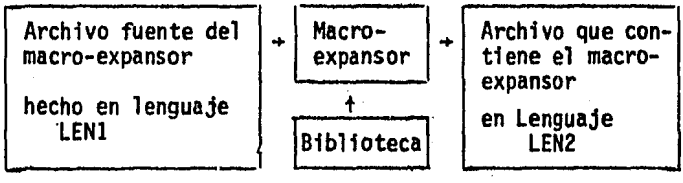


Fig. 2.1.

Esto se realizaría dando como archivo fuente al propio Macro-expansor, y así trasladar (transportar) el Macro-expansor de la máquina A a la máquina B.

b) Otra aplicación interesante es la llamada EDICION SISTEMATICA, lo cual se refiere a reemplazar todas las ocurrencias de un identificador dentro de un programa fuente, por otro identificador distinto.

La edición sistemática es utilizada cuando se desarrolla software por varias personas y donde cada una de las personas realiza un módulo de software. Sucede que al reunir todos los módulos existen identificadores que son repetidos o muy semejantes. Esto puede llevar a posibles malas interpretaciones al revisar todo el conjunto de módulos y tendería a ocasionar errores graves y muy difíciles de detectar.

c) La posibilidad de extender el lenguaje base, es decir la posibilidad de hacer, de un lenguaje base un lenguaje más poderoso. Como un ejemplo, nos podríamos referir al lenguaje Fortran, lo cual ciertamente no es un lenguaje muerto; su uso es muy generalizado y tiene muchas cualidades y para balancear sus múltiples deficiencias se creó un preprocesador o Macro-procesador llamado RATFOR, del cual hablaremos a continuación.

En general, se puede decir que a más alto nivel del lenguaje de programación se necesita menos de los Macros, esto es debido a que dichos lenguajes van teniendo nuevas estructuras y nuevas proposiciones de tal manera que tienden a facilitar al programador su trabajo.

Sin embargo, aún en los lenguajes de alto nivel más comprensibles y completos, los Macros son utilizados para introducir proposiciones especialmente diseñadas para la aplicación particular del usuario.

Podemos ver el lenguaje Fortran, el cual es muy usado para resolver problemas de Análisis Numérico, para este caso específico se creó MPAN (Macro-procesador de Análisis Numérico), el cual además extiende el lenguaje Fortran a uno más poderoso por la utilización de funciones iterativas (REPEAT-UNTIL, WHILE-ENDW, FOR-ENDFOR).

2.2. El Macro-procesador RATFOR

RATFOR es un preprocesador de un tamaño significante. La ventaja de un preprocesador es que no se tiene que escribir un compilador para obtener un mejor lenguaje; en lugar de eso se construirá en el trabajo de otros.

El propósito primordial de RATFOR es hacer de Fortran un mejor lenguaje de programación; para escribirlo y explicarlo, por permitir programas redactables y bien estructurados. La estructura que controla el flujo de RATFOR son las instrucciones if-else, while, do,break, next y oraciones agrupadas por paréntesis. Estas estructuras son enteramente adecuadas para programar sin GO TO.

El aspecto cosmético de RATFOR ha sido designado a hacer conciso y agradablemente razonable a los ojos. Este preprocesador es de forma libre: las instrucciones pueden aparecer en cualquier lugar de la línea de entrada. El fin de una línea generalmente marca el fín de una oración pero las líneas que obviamente no han terminado se colocan una "," al fin de la línea, automáticamente continua a la siguiente línea.

Varias instrucciones pueden aparecer en una línea si están separadas por ";". El signo "#" en cualquier parte de la línea significará el empiezo de un comentario.

El procesado está organizado como sigue. La parte central del - - -

preprocesador es la rutina que analiza la estructura gramatical de la entrada. Al principio de cada instrucción, esta rutina llama a otra rutina "analizador léxico" para clarificarlo dentro de uno de los tipos especificados en la gramática. Como el preprocesador no utiliza ningún caracter de advertencia, el "analizador léxico" tiene que revisar palabra por palabra el registro de entrada. Este preprocesador además utiliza una Biblioteca de Macro, para mayor información ver (Levine).

2.3. Diferencias entre RATFOR y MPAN

MPAN es un preprocesador o Macro-procesador de propósito particular al igual que RATFOR. Los dos Macro-procesadores tienen en común el de expander el lenguaje Fortran, es decir, hacer un lenguaje Fortran más estructurado, utilizando funciones iterativas.

Aunque los dos tienen la misma finalidad, en el área de la estructuración en Fortran, MPAN además de manejar una Biblioteca de Macros como RATFOR, también maneja una Biblioteca de Análisis Numérico, ya que MPAN fue creado como un lenguaje que ayudara al usuario a resolver problemas de Análisis Numérico.

Aunque RATFOR y MPAN a simple vista parecen ser muy similares, existen algunas diferencias entre ellos.

MPAN es un preprocesador en modo de advertencia porque utiliza un carácter de señal, mientras que RATFOR es un preprocesador de modo libre, porque no utiliza ningún carácter de señal. Para los usuarios de un preprocesador o Macro-expansor es más cómodo utilizar un preprocesador en modo libre, ya que así no necesita tener cuidado con los caracteres de advertencia. Esto que parece una ventaja a simple vista, al

realizarse el proceso se convierte en una gran desventaja ya que cada identificador del programa fuente tendrá que ser buscado en las tablas de palabras reservadas del preprocesador. Todo esto desemboca en un mayor tiempo de procesado.

Mientras que el procesador de modo de advertencia permite reconocer rápidamente si el texto es parte del lenguaje base o una palabra reservada de MPAN.

Tanto MPAN como RAFTOR, tienen bien definido su caracter de comentario. Esto es algo importante ya que el preprocesador cuando detecta el caracter de comentarios pasa el texto al archivo de salida, tal cual sin perder tiempo en revisarlo.

Tanto MPAN como RAFTOR, tienen una rutina central que analiza el texto fuente por átomos. Esta característica es importante, ya que existen preprocesadores que manejan el texto fuente caracter por carácter y cuando hay algún cambio, ya sea de caracter de señal (por ejemplo) es necesario hacer un cambio en varias partes del preprocesador, mientras que en los preprocesadores que manejan el texto fuente por átomos, con sólo hacer la corrección en la rutina que analiza, se corregirá el problema.

Resumiendo MPAN fue diseñado pensando en las características fundamentales, deseables en un Macro-procesador.

La Figura 2.1 nos muestra la diferencia entre RAFTOR y MPAN de una manera general.

Propósito	MPAN Particular	RATFOR Particular
Bibl. Macros	SI	SI
Otra Bibl.	SI	NO
Modo de trabajo	Advertencia	Libre
Detecta caracter comentarios	SI	SI
Forma de tratar el texto	Atomo	Atomo

Fig. 2.1

2.4. Conclusión

MPAN (Macro-procesador de Análisis Numérico), tiene mayor utilidad que RAFTOR por contar con la Biblioteca de Análisis Numérico además de la de Macros. Además de contar con un modo de trabajo que ocupa menos tiempo de procesado.

III. DESCRIPCION DEL MACRO-PROCESADOR DE ANALISIS NUMERICO

3.1. Introducción

En este capítulo se describirá el Macro-procesador de Análisis Numérico, su modo de trabajo, la forma como maneja el texto.

Se describirá cada archivo de entrada y salida a MPAN, así como los archivos temporales (utilizados nada más en el proceso).

Se verá la manera como MPAN trabaja las instrucciones que se encuentran en el programa del usuario, el cual estará compuesto de: instrucciones del lenguaje base, definiciones de Macros, Macro-llamadas, anidación de llamadas a subrutinas en el cuerpo del Macro, las proposiciones de asignación directa y las funciones predefinidas.

Del programa del usuario MPAN generará un programa en lenguaje base.

3.2. Modo de trabajo de MPAN (Macro-procesador de Análisis Numérico)

MPAN trabaja con un caracter de advertencia (también llamado de señal), que el preprocesador reconoce rápidamente si el texto es parte del lenguaje base o una palabra reservada de MPAN. Este caracter de advertencia (señal) permite que el tiempo de procesado sea menor ya que únicamente se tomará en cuenta las palabras del texto fuente que empiecen con dicho caracter, mientras que las otras serán pasadas al archivo tal cual.

El caracter de advertencia (señal) en MPAN será "%".

A los procesadores que utilizan un caracter de señal se les conoce como preprocesador de modo de advertencia. Existen preprocesadores llamados de modo libre, que no necesitan ningún caracter de señal.

3.3. Forma de manejo del texto

Existen dos formas de tratar el texto fuente (programa escrito por el usuario) ya sea por caracteres o átomos. El tratar el texto por caracteres consiste en ir leyendo del archivo fuente caracter por caracter y así irlos examinando, de tal forma que dichos caracteres serán pasados tal cual al archivo de salida, a menos que el caracter leido sea un caracter especial (advertencia). En el momento de reconocer el caracter especial, se van leyendo los caracteres subsecuentes para determinar la palabra reservada.

La forma de tratar el texto llamado por átomos consiste en leer el texto fuente caracter por caracter y construir elementos constituidos por caracteres válidos ya sea para palabras reservadas o para identificadores y números. Estos átomos son construidos por un analizador léxico (scanner), de tal forma que el análisis del texto sea sobre dichos átomos MPAN trabaja en base a átomos ya que en el modo de caracter por caracter el propio preprocesador tiene que examinar el texto fuente. Este examen del texto hecho por el preprocesador no permite modificaciones fáciles, ya que si se desea cambiar la estructura de los identificadores (cambiar por un lenguaje base extraño) habría que modificar el propio preprocesador en varios de sus módulos. Pero usando el modo de átomos, la modificación consistirá únicamente en cambiar el analizador léxico.

Algo que hay que mencionar a favor del modo de caracter por caracter es la velocidad. Este modo es más rápido que el de átomos, ya que no se pierde tiempo en la construcción de átomos.

Una característica muy importante y poderosa de MPAN es la de poder reconocer cuando el texto leído es parte de un comentario. Esta característica en los Macro-procesadores de Propósito General es imposible, debido a que cada lenguaje maneja sus comentarios en formas diversas, por ejemplo: en los lenguajes ensambladores, el caracter de comentarios comúnmente es el caracter ";", mientras que en otros lenguajes como Fortran su carácter de comentarios se encuentra en la primera columna de cada registro y será "C" si el registro es un comentario.

En los Macro-procesadores de propósito específico, como lo es MPAN, en el cual se está construyendo un archivo en lenguaje Fortran y tiene de entrada un programa en Fortran es fácilmente localizable el caracter de comentario y de esa manera no es necesario examinar el texto fuente que corresponda a comentarios, ayudando en el ahorro de tiempo de procesado, ya que cuando se ha detectado que el texto es comentario, todo el texto fuente es pasado directamente al archivo de salida y este proceso se detiene hasta que se encuentre el caracter de fin de comentario.

3.4. Macro-procesador de Análisis Numérico

El Macro-procesador de Análisis Numérico (MPAN) trabaja en la base de ser un preprocesador, es decir, sigue la línea de la mayoría de los Macro-procesadores. MPAN es un módulo de software separado de cualquier traductor y el texto antes de entrar a los traductores debe haber pasado por el Macro-procesador MPAN, por esto es el adjetivo de preprocesador.

Según la Fig. 3.1, MPAN utiliza cuatro archivos diferentes:

- El archivo donde se encuentra la Biblioteca de Análisis Numérico.
- El archivo donde se encuentra el programa fuente, el programa escrito por el usuario.
- Un archivo auxiliar donde se encuentran los Macros, ya sean, los definidos por el usuario o los de Biblioteca.
- Un archivo de salida que contendrá un programa fuente en Fortran.

Este programa que se encuentra en el archivo de salida, a través de los comandos que el usuario tecleará desde su terminal (FOR <nom. prog.> = <nom. prog> y TKB <nom. prog.> = <nom. prog.>), entrarán al compilador de Fortran, el cual creará un programa objeto ejecutable.

Al terminar este proceso el usuario correrá (o dará RUN <nom.prog.>).

3.4.1. Archivo de la Biblioteca

Este archivo contiene una serie de subrrutinas con un propósito bien específico; en este caso particular, la Biblioteca de Análisis Numérico estará formada por una serie de subrrutinas o funciones del Algebra Lineal usados en el LINPACK, que en este momento es una de las herramientas más poderosa con que cuenta el Análisis Numérico para la resolución de Métodos del Algebra Lineal Numérica.

Las subrrutinas o funciones de esta Biblioteca se encuentran en lenguaje Fortran.

La finalidad de tener una Biblioteca es la de darle al usuario --

herramientas poderosas para la resolución de sus problemas de Métodos Numéricos, es por esta razón por la cual se escogió las subrrutinas del LINPACK.

MPAN (Macro Procesador de Análisis Numérico)

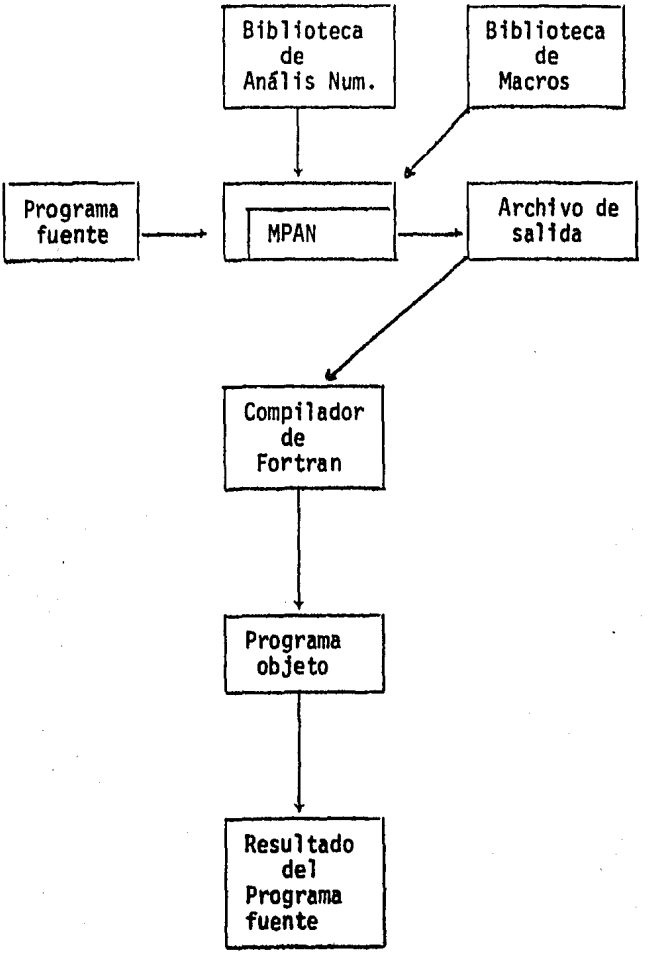
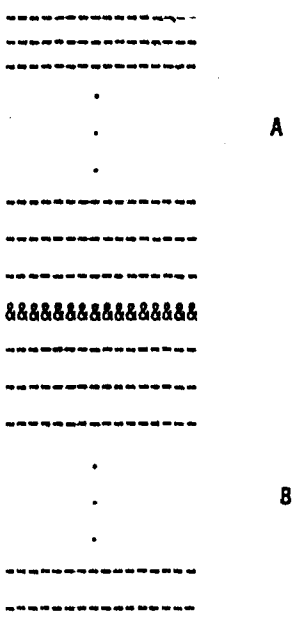


Fig. 3.1

El archivo de la Biblioteca de Análisis Numérico está estructurado de la siguiente manera:



En la parte A se encuentran los nombres de las subrrutinas y los nombres de las subrrutinas llamadas por ellas (si es que existen) además una liga que será la dirección donde se encuentra el cuerpo de la subrrutina, así como un valor numérico que nos indicará, si se trata de una subrrutina (1) o si se trata de una función (0).

En B se encuentran los cuerpos de las subrrutinas o funciones de Análisis Numérico.

3.4.2. El archivo auxiliar

En memoria secundaria tenemos un archivo llamado "auxil1.arc" que está estructurado en tres partes.

i) Contiene nombres de los Macros, la dirección donde se encuentra el cuerpo del Macro y la longitud del cuerpo del Macro.

ii) Contiene nombres de las Metavariables utilizadas en los Macros y el valor de la Metavariable.

iii) Contiene el cuerpo de los Macros.

Los Macros están estructurados por capas, de la siguiente manera: a los Macros que se encuentran en la primera capa, se les llama Macros primitivos, a los Macros que se encuentran en la segunda capa están construidos en base a los Macros primitivos (primera capa) y así los Macros que se encuentran en capas superiores (segunda, tercera, enésima capa) llaman o están construidos en base a las capas anteriores.

La Figura 3.2, simboliza las capas que estructuran los Macros dentro de la Biblioteca de Macros.

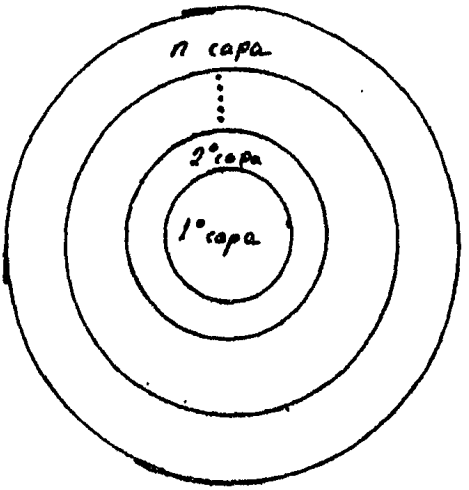


Fig. 3.2

MPAN maneja al archivo auxill.arc de la siguiente manera:

Abre el archivo auxill.arc y pasa la primera parte a una tabla de Macros, pasa la segunda parte (ii) a una tabla de Metavariables y copia la tercera parte (iii) al archivo auxiliar, quedando en este archivo únicamente los cuerpos de los Macros estructurados por capas y es con este archivo con el cual trabaja MPAN.

Si el usuario decide hacer declaraciones de Macros en su programa fuente, MPAN se encargará de pasar los Macros definidos del programa fuente, al archivo auxiliar.

3.4.3. Archivo que contiene el programa fuente

El programa que se encuentra en este archivo fue escrito por el usuario con un lenguaje que llamaremos "seudo-fortran" porque respetará la estructura del lenguaje Fortran e incluso utilizará algunas de sus instrucciones, en él encontraremos además:

a) Definiciones de Macros
b) Anidación de Macros
c) Macro-llamadas
d) Llamadas a subrrutinas
e) Anidación de Macro-llamadas
f) Anidación de llamadas a subrrutinas en el cuerpo del Macro
g) Proposiciones de asignación directa
h) Funciones predefinidas
i) Instrucciones del lenguaje base

MPAN maneja cada uno de estos incisos de forma diferente, lo cual se describirá a continuación.

3.4.3.1. Definiciones de Macros

MPAN trabaja en modo de advertencia y para definir un Macro se escribe:

```
%MACRO NOMBRE (ARG1, ARG2, ..., ARGN)
 CUERPO DEL MACRO
%ENDM
```

En donde se podrán notar las palabras reservadas %MACRO y %ENDM, las cuales están precedidas por el caracter de advertencia "%".

Al definir un Macro (%MACRO), el nombre de dicho Macro (NOMBRE) y los argumentos de dicho Macro (ARG1, ARG2, ..., ARGN), serán almacenados en una tabla formada por nombres de Macros y argumentos del Macro.

El CUERPO DEL MACRO será almacenado en el archivo auxiliar, donde permanecerá, para poder ser accesado cuando se haga un llamado a dicho Macro. El CUERPO DEL MACRO no será almacenado tal cual fue puesto por el usuario, sino que, el preprocesador (MPAN) sustituirá (en el cuerpo del Macro) los parámetros formales por un caracter de advertencia (señal) y un número. El número dependerá de la posición del parámetro formal dentro de la lista de parámetros formales (argumentos) dados en la definición.

MPAN utiliza en los parámetros formales como ya se describió el caracter "∿", este caracter de señal es distinto al usado para determinar las palabras reservadas o pseudo-instrucciones (%), ya que en caso contrario podría existir la posibilidad de confusión en el preprocesado. Este nuevo caracter de advertencia no debe ser del conjunto de caracteres válidos para la aplicación del Macro-procesador.

El argi equivale a "∿" en el cuerpo del Macro en donde "∿" es el caracter de advertencia e "i" es la i-ésima posición dentro de la lista de argumentos formales (a "∿i" le llamaremos argumentos de advertencia número i).

La sustitución de parámetros formales por argumentos de advertencia permitirá una mejor y más rápida sustitución. La justificación es la siguiente:

a) El proceso se hace más rápido (o sea la sustitución de los parámetros actuales por los argumentos de advertencia) debido a que el procedimiento que sustituye busca únicamente los caracteres de advertencia, y todo el texto es pasado directamente sin ser examinado. Si se piensa en la posibilidad de no usar caracteres de advertencia entonces se tendría que ir preguntando por cada identificador que fuera parte del cuerpo del Macro para ver si es o no parámetro formal, lo cual evidentemente se resume en más tiempo de procesado (este esquema es un caso particular del paso de parámetros por palabra clave).

b) El uso de argumentos de advertencia es mejor en el sentido de que implica menor posibilidad de error para el usuario debido a que el cuerpo del Macro estando ya definido no contiene los identificadores tal cuales fueron definidos (contienen los argumentos de advertencia), de tal forma el usuario puede utilizar como parámetro actual un identificador que haya sido utilizado como parámetro formal.

Las Macro-definiciones son terminadas con la palabra %ENDM como se muestra en el ejemplo anterior.

3.4.3.2. Anidación de Macros

Un punto importante de MPAN es la de admitir Macro-definiciones dentro de Macro-definiciones. Esto es, que en el cuerpo de una Macro-definición existiera otra Macro-definición, la cual sería "propia" de la que contiene, un ejemplo sería:

```
 %MACRO M1 (X)
--------------
--------------
--------------
 %MACRO M2 (Y)
--------------
--------------
--------------
 %ENDM
--------------
--------------
--------------
 %ENDM
```

Al efectuarse un llamado del Macro M1 (externo) se obtiene la definición de M2 (interno). Esto es debido a que el cuerpo de una Macro-definición es pasado tal cual al archivo auxiliar, es decir, no se hace ningún tipo de análisis.

Cuando el cuerpo del Macro (externo) está en el archivo auxiliar (temporal) mantiene la Macro-definición que contiene dentro de su cuerpo y así al efectuarse un llamado al Macro-externo, se toma el cuerpo del Macro y se va expandiendo, pero si en el texto que se va expandiendo hay una Macro-definición, se le toma en cuenta para poder ser expandido como cualquier Macro-definición.

Esta característica del MPAN será realizada en primera instancia en la secuencia de que por cada llamada al Macro externo se efectúe una definición del Macro, permitiendo mayor flexibilidad al usuario. Algo importante es que no se puede llamar al Macro interno dentro del Macro externo, porque MPAN trata la Macro-definición interna como cualquier Macro definición pasándola al archivo auxil2 la Macro-definición interna, la cual cuando acaba de expander al Macro-externo, le une al archivo auxiliar el archivo auxil2.

3.4.3.3. Macro-llamadas

Para realizar una Macro-llamada (también conocida como Macro-expansión) es necesario llamar al Macro por su nombre utilizando el caracter de advertencia de la siguiente forma:

%NOMBRE(A, B, C, ..., Z)

El identificador para el nombre, así como los identificadores comunes, serán de la forma: el primer carácter será alfabético seguido de caracteres alfanuméricos.

La longitud de los identificadores está determinada por la constante MAXLON.

En el llamado al Macro se dan los parámetros actuales (A, B, ...,Z) y la relación de los parámetros actuales con los parámetros formales es posicional, es decir, el primer parámetro actual corresponde al primer parámetro formal; el segundo parámetro actual corresponde al segundo parámetro formal y así sucesivamente.

Si los parámetros actuales son menor en número que los parámetros formales, entonces los parámetros formales restantes tendrán asignado

el valor nulo. También es válido que el usuario pase como parámetro actual la cadena nula. Hay que hacer notar que no importa si es una Macro-llamada en la cual hay más parámetros actuales que parámetros formales ya que MPAN ignora a los parámetros actuales sobrantes.

Si alguno de los parámetros actuales consiste de un conjunto de caracteres conteniendo cualquier tipo de separador o caracteres especiales entonces deberá ser colocado dentro paréntesis cuadrados.

Por ejemplo, si el parámetro actual es I = I + 1 entonces un llamado podría ser:

%NOMBRE (A, B, [I = I + 1], D)

Igualmente, es posible que un parámetro actual contenga entre sus caracteres el paréntesis cuadrado, por ejemplo, supóngase la siguiente Macro-llamada:

%NOMBRE (A, B, [IF A[I]> 0 THEN] , D, E)

Esto es permitido por MPAN ya que toma los paréntesis cuadrados por niveles, de tal manera, que en la Macro-expansión no toma en cuenta los paréntesis cuadrados del primer nivel, como se ilustra en la Fig. 3.3

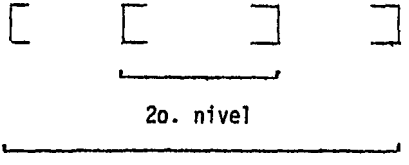


Fig. 3.3

3.4.3.4. Llamadas a subrrutinas o funciones

Para realizar una llamada a una subrrutina, se hace de la siguiente manera:

%NOMBRE (A, B, ..., Z)

Para realizar una llamada a una función, se hace de la siguiente manera:

VAL = %NOMBRE (A, B, ..., Z)

Como se dijo anteriormente, llamar a una subrrutina es llamar a un proceso específico que nos modificará en algo nuestras variables, mientras, que llamar a una función es llamar a un proceso que además de modificar nuestras variables retornará un valor. Ese valor será asignado a alguna variable, en este caso VAL.

Nótese que el llamar a una subrrutina, función o Macro se hace de la misma manera y la única manera como MPAN hace la diferencia es revisando las tablas que contienen los nombres de las subrrutinas o funciones y la tabla que contiene los nombres de los Macros.

Cuando en el programa fuente se realiza una Macro-llamada el MPAN lo expanderá en ese preciso instante, mientras que si lo que se realizó fue una llamada a subrrutina o función, MPAN pasa al archivo de salida el nombre de la subrrutina o función con sus argumentos y dependiendo, si es subrrutina le antepone la palabra CALL, si es una función no le antepone nada. Después busca el nombre de la subrrutina o función en la tabla que está formada con los nombres de las subrrutinas o funciones, que en ese programa fuente se están utilizando. Si el nombre no se encuentra en la tabla se da de alta en la tabla, con la dirección donde

se encuentra el cuerpo de la subrrutina o función y se le asigna un valor (0 ó 1) dependiendo, si el nombre corresponde a una función o a una subrrutina respectivamente.

Cuando llega al fin del archivo del programa fuente entonces MPAN, toma la tabla con los nombres de las subrrutinas que se ocupan en ese programa, las busca en el archivo de la Biblioteca y las pasa al archivo de salida.

3.4.3.5. Anidación en Macro-llamadas

El MPAN permite anidación en las llamadas a Macros, es decir, que dentro del cuerpo de un Macro, se llame a otro Macro, por ejemplo:

```
%MACRO M1 (X)
 IF X >0 THEN X = X - 100
%ENDM
%MACRO M2 (Y)
 A = SEN (Y)
%M1 (A)
%ENDM
 BEGIN
%M2 (30)
 END.
```

MPAN, analiza el programa fuente, guardará en su archivo temporal, las dos definiciones de Macros que se encuentren en su programa fuente sin revisar si existen llamadas a otros Macros dentro de los cuerpos de los Macros.

```
%MACRO M1 (∿1)
 IF∿1>0 THEN ∿1 = ∿1 - 100
%ENDM
%MACRO M2 (∿1)
 A = SEN (∿1)
%M1 (A)
%ENDM
```

Sólo en el momento que un Macro es llamado, al irlo expandiendo, se analizará el contenido del cuerpo del Macro, si dentro de un Macro se llama a otro Macro MPAN, expanderá el nuevo Macro llamado y sólo cuando se terminó la expansión del Macro llamado internamente, proseguirá con la Macro-expansión que había sido detenida.

Con lo cual en el archivo de salida quedará así:

```
BEGIN
A = SEN (30)
IF A > 0 THEN A = A - 100
```

De esta manera no importará el orden en que el usuario defina sus Macros, ya que todos quedarán guardados en un archivo temporal, y sólo cuando sean llamados, se revisará la condición de si existen llamadas a otros Macros, ya que se encuentran en el archivo temporal.

3.4.3.6. Anidación de llamadas a subrrutinas o funciones en el cuerpo del Macro

El MPAN, permite anidación en las llamadas a subrrutinas, es decir, que dentro del cuerpo de un Macro se llame a una subrrutina o función que se encuentre en la Biblioteca de Análisis Numérico.

```
%MACRO M1 (X, Y)
 A = SEN (X)
 B = SEN (Y)
%MULTIPLY (A, B)
%ENDM
 BEGIN
%M1 (10, 15)
 END.
```

Al igual que en la anidación de Macros, cuando el usuario, define Macros en su programa fuente, MPAN, guarda esos Macros definidos en un archivo temporal, sin revisar el contenido del cuerpo del Macro.

```
%MACRO M1 (∿1,∿ 2)
 A = SEN (∿1)
 B = SEN (∿2)
%MULTIPLY (A, B)
%ENDM
```

Sólo en el momento que un Macro es llamado al expanderse, se analizará el contenido del cuerpo del Macro, si dentro de un Macro se llama a una subrrutina o función, MPAN, pasará al archivo de salida el nombre de la subrrutina o función con sus argumentos, si se trata de una subrrutina le antepondrá la palabra CALL, sino la pasará tal cual, con lo cual el archivo de salida quedará así:

```
BEGIN
A = SEN (10)
B = SEN (15)
CALL MULTIPLY (A, B)
END.
```

3.4.3.7. Proposiciones de asignación directa

En el archivo auxil1.arc, se encuentran las proposiciones de asigna ción directa o metavariables usados en el cuerpo de los Macros, que al principio del proceso de MPAN se crea la tabla de proposiciones de asignación directa (metavariables) con sus valores.

MPAN admite proposiciones de asignación directa, más específicamen te una proposición de asignación directa permite igualar un símbolo a un valor específico.

Cuando una proposición de asignación directa (metavariable) es uti lizada y el símbolo involucrado aparece por primera vez, el símbolo es colocado dentro de la tabla de metavariables y se le asigna el valor 0.

Un símbolo que se encuentra en la tabla puede ser redefinido por una subsecuente proposición de asignación directa, por asignarle un nue vo valor al símbolo previamente definido.

El formato general de una proposición de asignación directa es:

. %SIMBOLO = EXPRESION

En donde SIMBOLO es una cadena de caracteres alfanuméricos y el pri mer carácter debe ser alfabético, además el símbolo debe ser precedido por el carácter de advertencia.

Las EXPRESIONES son combinaciones de términos unidos por operadores binarios, las cuales son reducidas a un solo valor, es decir, son evaluadas. Los términos deben ser de tipo entero.

Un término es un componente de una expresión y puede ser un número entero, un símbolo que se encuentre en la tabla de Metavariables o bien un símbolo que no se encuentra definido en la tabla de Metavariables.

Si el término es un símbolo que se encuentra en la tabla de Metavariables, entonces se toma su valor correspondiente de la tabla de Metavariables. Y si el término es un símbolo que no se encuentra en la tabla, toma el valor de cero.

Las expresiones son evaluadas de izquierda a derecha sin reglas de precedencia sobre los operadores, excepto que los operadores unarios toman precedencia sobre operadores binarios. Un término precedido por un operador unario es considerado a contener ese operador.

Un término o expresión perdida será interpretado como un cero. Un error en los operadores o en los términos, terminará el análisis de la expresión. Por ejemplo, la expresión.

```
20 + 10        500
```

es evaluada como (20 + 10) ya que el primer carácter distinto de blanco, que está después del número 10, no es un operador legal (+, -, x, /).

Los espacios dentro de las expresiones son ignorados.

Ejemplo de proposiciones válidas:

```
%CONTADOR = 3
%INDICE =  %BASE x 3 + 5
```

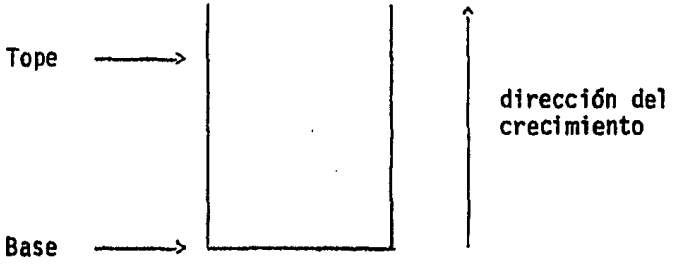
3.4.3.8. Funciones predefinidas

MPAN tiene funciones predefinidas implícitas, es decir, funciones que son parte de MPAN.

En esta sección se describirán las funciones predefinidas PUSH y POP.

Debido a la amplia aplicación de MPAN se pensó en darle al usuario la posibilidad de manejar una estructura de datos en tiempo de procesado. Y pensando en la estructura más acorde a las aplicaciones de MPAN se construyó una pila. Una pila es una lista lineal que se caracteriza porque el primer elemento en entrar a la lista es el último en salir.

Gráficamente la pila se ve de la siguiente manera:



Como se podrá notar las inserciones y las supresiones se hacen por el TOPE. En el siguiente capítulo (REALIZACION) se describirá completamente la estructura, ya que en esta sección se verá la forma de utilizar las funciones únicamente.

La pila será manejada por el usuario haciendo llamadas a las funciones predefinidas que son:

PUSH. Mete el valor del primer símbolo que haya sido definido por el usuario a la pila y actualiza el apuntador a dicha pila.

POP. Actualiza el apuntador y saca el valor que está en el tope de la pila para depositar como el valor del segundo símbolo que fue definido por el usuario.

El uso de esta estructura permitirá al usuario anidación de proposiciones o unidades sintéticas definidas por él mismo. Hay que hacer --

notar que el manejo de la pila será hecho por el usuario en tiempo de procesado.

3.4.3.9. Instrucciones del lenguaje base

El programa que escribirá el usuario está formado por dos tipos de lenguajes, uno que llamaremos el lenguaje base, que como se dijo al principio son instrucciones de Fortran y otro el lenguaje fuente por contener instrucciones de los descritos en los incisos anteriores.

Estas instrucciones del lenguaje base, MPAN las reconoce por no estar precedidas por el caracter de advertencia "%" y las pasa al archivo de salida tal cual. Es obligación del usuario, que estas instrucciones estén bien construidas, ya que MPAN no las revisa sintácticamente.

3.4.4. Archivo de salida

MPAN creará el archivo de salida con el nombre salida.ftn que estará compuesto por un programa en Fortran:

Este contendrá instrucciones de Fortran, llamadas a subrrutinas y las subrrutinas correspondientes.

Este archivo "salida.ftn", se compilará, ligará y correrá de la siguiente manera:

```
FOR SALIDA = SALIDA
TKB SALIDA = SALIDA
RUN SALIDA
```

Y de esta manera, obtendremos los resultados de algún problema de Análisis Numérico.

3.5. Conclusión

Este Macro-procesador de Análisis Numérico, al igual que la mayoría de los Macro-procesadores siguió la misma línea: como es la de contar con anidación de Macros, llamada a Macros dentro del cuerpo del Macro, expansión de Macros, guarda en memoria los Macros creados por el usuario etc.

IV. MPAN COMO UNA HERRAMIENTA PARA EL ANALISIS NUMERICO

4.1. Introducción

En este capítulo se verá lo útil que es el Macro-procesador de Análisis Numérico como una herramienta para resolver 3 problemas del Análisis Numérico: la Multiplicación de las Matrices, la inversa de la Matriz A (A^{-1}) y el último método será el de resover AX = B por el método iterativo de GAUSS-SEIDEL. Se escogieron estos 3 ejemplos por ser de mediana complejidad, se ordenaron del más sencillo, como es la Multiplicación de Matrices que además es un ejemplo típico, al más complicado como es la resolución de la ecuación AX = B por el método iterativo GAUSS-SEIDEL. Este último ejemplo es un programa que tiene tanto instrucciones del lenguaje base como instrucciones del lenguaje fuente.

Al tratar de resolver estos tres problemas, nos daremos cuenta de la importancia de las Bibliotecas que maneja MPAN.

MPAN maneja dos Bibliotecas importantes que son:

- La Biblioteca de Análisis Numérico, que contiene un conjunto de subprogramas usados en el LINPAK.
- La Biblioteca de Macros.

El usuario al tratar de resolver sus problemas, forzosamente necesita saber con qué Macros, subrrutinas o funciones cuenta. Por este motivo, en este capítulo se describirán cada una de las Bibliotecas y la

función de cada una de las Macros, subrrutinas o funciones, así como qué tipos de argumentos manejan éstas.

La metodología seguida en la resolución de estos problemas formará una mejor idea de cómo manejar las Bibliotecas.

Al final de la tesis se encontrarán el Anexo 3 y el Anexo 2 que contendrán las Bibliotecas de Análisis Numérico y Macros respectivamente.

4.2. Bibliotecas que maneja MPAN

Como se ha dicho anteriormente MPAN es un Macro-procesador de propiedad particular (para el Análisis Numérico) y esto se debe a sus Bibliotecas, MPAN maneja dos Bibliotecas:

A) Biblioteca de Análisis Numérico

B) Biblioteca de Macros

4.2.1. La Biblioteca de Análisis Numérico

Es un conjunto de funciones y nombres de subprogramas del Algebra Lineal Básico usado en el LINPACK: [DONGARRA]

i) Producto de Vectores

$$SW = \%SDOT\ (N, SX, INCX, SY, INCY)$$

$$SW = \sum_{i=1}^{N} X_i Y_i$$

Si N<=0, SDOT retornará con el valor 0.

ii) Operación Elemental de Vectores

$$\%SAXPY\ (N, SA, SX, INCX, SY, INCY)$$

$$Y = AX + Y$$

Si A = 0 ó si N<= 0 esta subrrutina regresa inmediatamente.

iii) Construcción de una rotación plana (de Givens)

%SRTG (SA, SB, SC, SS)

Dados A y B

$$\sigma \begin{cases} \text{sgn}(A) & \text{si } |A| > |B| \\ \text{sgn}(B) & \text{si } |B| \geq |A| \\ R = (A^2 + B^2)^{1/2} \end{cases}$$

$$C \begin{cases} A/R & \text{si } R \neq 0 \\ 1 & \text{si } R = 0 \end{cases}$$

$$S \begin{cases} B/R & \text{si } R \neq 0 \\ 0 & \text{si } R = 0 \end{cases}$$

Los números C, S y R satisfacen la ecuación Matricial:

$$\begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

La introducción de σ no es esencial para el cálculo de la Matríz de Givens, pero es necesario si después se quiere reconstruir C y S a partir de un número . Para este propósito la subrrutina también calcula:

$$Z \begin{cases} S & \text{si } |A| > |B| \\ 1/C & \text{si } |B| \geq |A| \text{ y } C \neq 0 \\ 1 & \text{si } C = 0 \end{cases}$$

La subrrutina regresa R en A y Z en B así como C y S.

Si el usuario quiere reconstruir C y S a partir de Z lo puede hacer de la siguiente forma:

Si $|Z| = 1$ entonces $C = 0$ y $S = 1$
Si $|Z| < 1$ entonces $C = (1 - Z^2)^{1/2}$ y $S = Z$
Si $|Z| > 1$ entonces $C = 1/Z$ y $S = (1 - C^2)^{1/2}$

iv) Aplicación a una rotación plana

%SROT (N, SX, INCX, SY, INCY, SC, SS)

$$\begin{bmatrix} Xi \\ Yi \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} = \begin{bmatrix} Xi \\ Yi \end{bmatrix} \text{ Para } i = 1, \ldots, N$$

Si N<=0 ó si C = 1 y S = 0 la subrrutina regresa inmediatamente

v) Copia el Vector X a Y

%SCOPY (N, SX, INCX, SY, INCY)

Y : = X

Si N<=0 regresa inmediatamente

vi) Intercambia los Vectores X y Y

%SSWAP (N, SX, INCX, SY, INCY)

X ——> Y Y —>X

Si N<=0 regresa inmediatamente

vii) Función que nos da la Norma Euclidiana de un Vector

SW = %SNRM2 (N, SX, INCX)

$$SW = \left[\sum_{i=1}^{N} |Xi|^2 \right] 1/2$$

Si N<=0 SNRM2 regresa con el valor 0

viii) Suma las Magnitudes de los Componentes de un Vector

SW = %SASUM (N, SX, INCX)

$$SW = \sum_{i=1}^{N} |Xi|$$

Si N<=0 SASUM regresa con el valor 0

ix) Producto Escalar

%SSCAL (N, SA, SX, INCX)

X : = AX

Si N <=0 la subrrutina regresa inmediatamente

x) Encuentra el índice del máximo elemento de un vector

IMAX = %ISAMAX (N, SX, INCX)

Xi = max $\{|X_i| : i = 1, \ldots, N\}$

Si N <=0 la función ISAMAX regresa con valor 0

4.2.2. Biblioteca de Macros

La Biblioteca de Macros está compuesta por Macros que se encuentran en diferentes capas. En la primera capa se encuentran los Macros primitivos; los Macros que se encuentran en la segunda capa, dentro de su cuerpo, llaman a Macros de la primera capa; los Macros que se encuentran en la tercera capa llaman a Macros que se encuentran en la primera y segunda capa; los Macros que se encuentran en la cuarta capa llaman a los Macros que se encuentran en capas anteriores (3a., 2a., 1a.) y así sucesivamente.

Macros que se encuentran en la 1a. capa

Llamamos Macros primitivos a los Macros que se encuentran en la primera capa y como su nombre lo dice, son Macros que contienen las instrucciones más elementales de un lenguaje como Fortran.

Hacemos referencia a instrucciones más elementales de Fortran, como es asignar un valor a una variable, declarar Matrices, etc.

Estas Macros primitivas sirven como base para construir Macros más complejas.

- %GENETI () Se incrementan las Metavariables ETI y AUX en 1 (generador de etiquetas).
- %AUX1ETI () Le asigno a la Metavariable AUXI el valor de la Metavariable ETI.
- %AUX2AUX () Le asigno a la Metavariable AUX2 el valor de la Metavariable AUX.
- %ETIAUX () Le asigno a la Metavariable ETI el valor de la Metavariable AUX.
- %AUXAUX2 () Le asigno a la Metavariable AUX el valor de la Metavariable AUX2.
- %ETIAUX1 () Le asigno a la Metavariable ETI el valor de la Metavariable AUX1.
- %ETIAUX2 () Le asigno a la Metavariable ETI el valor de la Metavariable AUX2.
- %AUX3AUX () Le asigno a la Metavariable AUX3 el valor de la Metavariable AUX.

%DEFMAT (W1,W2, W3) Define una Matríz (DIMENSION W1 (W2, W3)).

%DEFVEC (W1, W2) Define un vector (DIMENSION W1 (W2)).

%INICIND (W1) Le asigno a la variable W1 el valor de 0 (W1 = 0).

%INICCONS (W1, W2) Le asigno a la variable W1 el valor W2 (W1=W2).

Macros que se encuentran en la segunda capa

Los Macros que se encuentran en esta capa llaman dentro de sus cuerpos a Macros primitivos (1a capa), es decir, son Macros más complejos en su estructura lógica, ya que realizan una función bien definida, dentro del lenguaje Fortran, como sería leer una variable, leer una Matríz,etc.

Algo importante de esta capa es la de contener funciones iterativas las cuales forman el maquillaje del Fortran. Estas Macros se basan en el uso del stack y Macros primitivos.

- %LEEV (W1, W2) Lee el vector W1, cuya dimensión es W2 (W1(W2)).
- %LEEVH (W1, W2, W3) Lee de la Matríz W1, el renglón W2, que varía desde 1 hasta W3.
- %ESCV (W1, W2) Escribe el vector W1, cuya dimensión es W2 (W1(W2)).
- %ESCVH (W1, W2, W3) Escribe de la Matríz W1, el renglón W2 que varía desde 1 hasta W3.
- %ESC (W1) Escribe el valor entero de W1 en 8 campos.
- %ESCF (W1) Escribe el valor real de W1 en 10 campos de los cuales 4 son decimales.
- %ESCLET (W1) Escribe el letrero W1, donde W1 es una cadena de caracteres entre paréntesis cuadrados ([ ]).
- %LEEI (W1, W2, W3, W4) Lee cuatro enteros de 8 campos cada uno.
- %LEEI1 (W1) Lee un entero de 8 campos.
- %LEEF (W1) Lee un real W1 de 10 campos de los cuales 8 son decimales.

- %REPEAT () El par "REPEAT-UNTIL" es una estructura iterativa
- %UNTIL (W1) de control que nos indican que las instrucciones a continuación se repetirán hasta que W1 sea verdadera.

- %WHILE (W1) El par "WHILE-ENDW" es una estructura iterativa
  %ENDW () de control la cual nos indica que las instrucciones a continuación se repetirán mientras W1 sea verdadera.

- %FOR (W1, W2, W3) El par "FOR-ENDFOR" es una estructura iterati
- %ENDFOR () va de control, la cual hará variar a W1 de 1 en 1 empezando en W2 hasta W3, cuando ya no se cumpla la condición (W1 W3), entonces se saldrá del FOR, por medio del ENDFOR.

Macros que se encuentran en la tercera capa

La construcción de los Macros que se encuentran en esta capa son un poco más complejos si los comparamos con los Macros de la 2a. capa, más bien tomamos como base a la 1a. y 2a. capa para construir tareas o funciones encaminadas al área de Análisis Numérico, esto significa, que en estas Macros, se encontrarán funciones como por ejemplo: comparar todos los elementos de un vector y encontrar el mayor. Todos estos Macros son sólo una pequeña parte de funciones que por lo general se nos presentan al resolver problemas de Métodos Numéricos.

- %LEEDIM (W1) Leemos el valor W1 y lo asignamos a N (número de incrementos).

- %ALIMMAT (W1, W2, W3) Leemos los valores de la Matríz W1 de W2 x W3(renglón por renglón).

- %LVECTOR (W1, W2) Leemos el vector W1 cuya dimensión es W2.

- %ESCMAT (W1, W2, W3) Escribimos los valores de la Matríz W1 de W2 x W3 (renglón por renglón).

- %NORMABS (W1, W2, W3) Dada la Matríz W1 de W2 x W3. Se encuentra en valor de i y j donde

$$X_{ij} = \max \{|X_{ij}| : i = 1, \ldots, W2 \quad j = 1, \ldots, W3\}$$

- %DIVELEM (W1, W2, W3, W4) Dada la Matríz W1, todos los elementos del renglón W2 los dividimos entre W4 (donde W3 es la dimensión de cada renglón de la Matríz W1).

- %LIMVEC (W1, W2) A todos los elementos del vector W1 cuya dimensión es W2 le asignamos el valor 0.0

- %MULTMAT (W1, W2, W3, W4, W5) Multiplica las Matrices W1 y W2, donde W1 está dimensionada por W3 x W4 y W2 por W4 x W5, deja el resultado de la Multiplicación en la Matríz C cuya dimensión es W3 x W5.

- %IDENTI (W1, W2) Dada la Matríz cuadrada (es decir W2 x W2) generamos la Matríz idéntica W1 = I.

- %MAYOR (W1, W2, W3, W4, W5, W6) Dada una Matríz W1 se comparan todos los elementos de la columna W6, dejando en W2 el valor absoluto del elemento más grande de esa columna y en W3 la posición. Todo esto está dentro de un FOR que revisa

los elementos desde W4 hasta W5.

- %INTERC (W1, W2, W3, W4) Dada una Matríz W1, intercambiamos dos de sus renglones donde W2 y W4 nos dicen la posición de los renglones a intercambiar y W3 es la dimensión de cada renglón.

- %OPERACION (W1, W2, W3, W4, W5, W6) Dada una Matríz W1, se realiza una operación entre dos de sus renglones, donde W4 y W5 nos dan las posiciones de los renglones que van a operar, W6 nos da el elemento de la diagonal del renglón W4 y a cada $A_{W4,j}$ del renglón W4 le va a restar $A_{W5,j} * A_{W4,W4}$ que se encuentran en el renglón W5. Toda esta operación se encuentra dentro de un FOR, donde W2 y W3 nos dicen de donde a donde varía el FOR respectivamente.

- %ITERA (W1, W2, W3, W4, W5) Dada la Matríz W1 y los vectores W2, W3 se calculan las sucesivas iteraciones X_i^{k+1} usando la siguiente fórmula:

$$X_i^{k+1} = A_{i,n+1} - \sum_{j=1}^{i-1} A_{ij}X_j^{k+1} - \sum_{j=i+1}^{W5} A_{ij}X_j^{k}$$

$$i = 1, \ldots, W4$$

Los Macros que se encuentran en la cuarta capa

En esta Biblioteca llegamos a construir Macros únicamente hasta esta capa, las cuales tienen como base los Macros de las capas anteriores.

Los Macros que se encuentran en esta Area son funciones que realizan operaciones complicadas de Análisis Numérico.

Se observa, por la forma como se fueron construyendo las capas, que no hay un límite para seguir construyendo capas, todo depende de la necesidad que se tenga en el Area de Análisis Numérico.

- %DIVDIAG (W1, W2, W3) Divide la i-ésima ecuación de la Matríz W1 por el elemento de su diagonal $A_{ii}(A_{ii} \neq 0)$, donde $A_{ij} = A_{ij} / A_{ii}$ $i = 1, \ldots, W2$; $j=1,\ldots,W3$

- %MAYDIAG (W1, W2, W3, W4) Dadas las Matrices W1, W2 cuyas dimensiones son W3 x W4 encontramos el elemento mayor de una columna de la Matríz W1 e intercambiamos los renglones en ambas Matrices.

- %DIVPIVOT (W1, W2, W3, W4, W5) Dadas las Matrices W1 y W2, se realizan las siguientes operaciones en ambas Matrices.
  Renglón W4 = Renglón W4 / $A_{w4,w4}$
  W3 = $A_{w4,w4}$* W3
  Donde W5 nos indica de 1 a que varía el renglón de la Matríz.

- %CONLINPIV (W1, W2, W3, W4) Dadas las Matrices W1, W2, se realiza la siguiente operación en ambas Matrices.
  Renglón W3 = Renglón W3 - $A_{w3,w4}$* Renglón W4.

Hasta aquí se describió una por una las Macros que integran la Biblioteca de Macros, así como las subrrutinas o funciones que integran la Biblioteca de Análisis Numérico, al final de la tesis se encontrarán en

el Anexo 2 la Biblioteca de Análisis Numérico y en el Anexo 3 la Biblioteca de Macros.

Maquillaje de Fortran

Uno de los objetivos de MPAN es la de estructurar el lenguaje Fortran, por medio de funciones iterativas, que controlen el flujo de las instrucciones, es decir, queremos darle al usuario funciones iterativas como las de Algol o Pascal.

Esas funciones iterativas a las cuales me refiero, las podemos encontrar en la Biblioteca de Macros, las cuales están compuestas de Macros primitivos y del uso de funciones predefinidas llamadas PUSH y POP, las cuales manejan un stack. Donde PUSH mete un elemento de la tabla de Metavariables (registro 1) al stack e incrementa su apuntador o tope en uno y POP decrementa el apuntador o tope en uno y saca del stack un elemento y lo guarda en la tabla de Metavariables (registro 2).

A continuación se dará una descripción breve de como estas funciones iterativas manejan el stack, quedando en el archivo de salida sólo instrucciones del lenguaje base.

a) REPEAT-UNTIL

```
%REPEAT                ETIQ1  CONTINUE                          A
 instrucción 1                 instrucción 1                    B
 instrucción 2                 instrucción 2                    C
%UNTIL (condición)             IF (.NOT.(condición)) GO TO ETIQ1 D
```

PASO 1 Cuando encuentra la Macro %REPEAT genera ETIQ1, hace un PUSH de ETIQ1 e imprime el renglón A.

PASO 2 Pasa las instrucciones B y C tal cual.

PASO 3 Cuando encuentra la Macro UNTIL, hace un POP, sacando ETIQ1 del stack e imprime el renglón D.

b) WHILE-ENDW

```
%WHILE (condición) ETIQ1 CONTINUE                          A
 instrucción 1          IF (.NOT.(condición)) GO TO ETIQ2   B
 instrucción 2          instrucción 1                       C
%ENDW                   instrucción 2                       D
                        GO TO ETIQ1                         E
                ETIQ2 CONTINUE                              F
```

PASO 1 Cuando encuentra el Macro %WHILE, genera ETIQ1, hace PUSH de ETIQ1 e imprime el renglón A, genera ETIQ2 e imprime el renglón B, hace un POP a ETIQ1, hace un PUSH a ETIQ2 y por último hace un PUSH a ETIQ1 quedando en el stack ETQ2 primero y ETIQ1 en el tope.

PASO 2 Pasa las instrucciones C y D tal cual.

PASO 3 Al encontrar la Macro %ENDW, hace un POP a ETIQ1 que se encuentra en el tope e imprime la instrucción E, hace un POP a ETIQ2 e imprime la instrucción F.

c) FOR-ENDFOR

```
%FOR (Cond1, cond2, cond3)   DO ETIQ1 cond1 = cond2, cond3   A
 instrucción 1               instrucción 1                   B
 instrucción 2               instrucción 2                   C
%ENDFOR ()                 ETIQ1 CONTINUE                   D
```

PASO 1 Cuando encuentra el Macro %FOR, genera ETIQ1, hace un PUSH a ETIQ1 e imprime el renglón A.

PASO 2 Pasa las instrucciones B y C tal cual.

PASO 3 Al encontrar la Macro %ENDFOR (), hace un POP a ETIQ1 e imprime el renglón D.

4.3. Multiplicación de Matrices

Según (Mc Call) el producto C = AB de dos Matrices A y B existe si el número de columnas de A es igual al número de renglones de B. El producto A (I x J) y B (J x K) es la Matríz C (I x K) con elementos C_{ik} definidos por:

$$C_{ik} = \sum_{j=1}^{J} A_{ij} B_{jk}$$

El diagrama de flujo para la Multiplicación de Matrices se encuentra en la Fig. 4.1.

El siguiente paso es programar en Fortran el programa para la resolución del problema de la Multiplicación de Matrices. Otra posibilidad sería la utilización de MPAN. La diferencia entre programar en Fortran o utilizar MPAN para la resolución de un problema cualquiera (en Análisis Numérico), es la siguiente:

i) Para programar en Fortran, se necesita primero conocer el lenguaje Fortran, así como saber diseñar el diagrama de flujo y saber pasar o traducir el diagrama de flujo a instrucciones de Fortran.

ii) Mientras que MPAN será una herramienta para llegar al mismo resultado, sin necesidad de conocer el lenguaje Fortran sólo será necesario revisar las Bibliotecas con que cuenta MPAN para saber con qué herramientas se cuenta.

Al revisar la Biblioteca de Macros encontramos los siguientes Macros que nos pueden ayudar a resolver el problema de Multiplicación de Matrices.

- DEFMAT (W1, W2, W3)

  La cual nos define una Matríz. Entonces para definir la Matriz A de 10 x 10 se sustituye W1 por A, W2 por 10 y W3 por 10. Podemos utilizar esa misma Macro para definir la Matríz B de 10 x 10 y la Matríz C de 10 x 10.

- LEEI (W1, W2, W3, W4)

  Con la cual podemos leer los valores I, J, K, sustituyendo W1 por I, W2 por J y W3 por K.

- ALIMMAT (W1, W2, W3)

  Con la cual podemos leer una Matríz si sustituimos W1 por A, W2 por I y W3 por J y la misma Macro nos sirve para leer la Matríz B de J x K.

- MULMAT (W1, W2, W3, W4, W5)

  Multiplica dos Matrices W1W2 donde W1 está dimensionada W3 x W4 y W2 está dimensionada W4 x W5 y nos dejaría el resultado en la Matríz C de W3 x W5. Como queremos multiplicar AB entonces sustituimos W1 por A, W2, por B, W3 por I, W4 por J, W5 por K y de esta manera obtendremos una Matríz C de I x K con el resultado de la multiplicación.

- ESCVH (W1, W2, W3)

  Esta Macro escribe o da los resultados de una Matríz W1 de W2 x W3, si queremos saber el resultado de la multiplicación y en C de I x K tenemos el resultado, entonces sustituimos W1 por C, W2 por I y W3 por K.

Con la utilización de estos Macros, resolvemos el problema de la multiplicación de Matrices. A continuación se mostrará el programa documentado -antes de entrar a MPAN-, el programa en Fortran -resultado de MPAN-, el cual es compilado, ligado y ejecutado de la siguiente manera:

```
FOR SALIDA = SALIDA
TKB SALIDA = SALIDA
RUN SALIDA
```

Obteniendo los resultados que también se muestran a continuación.

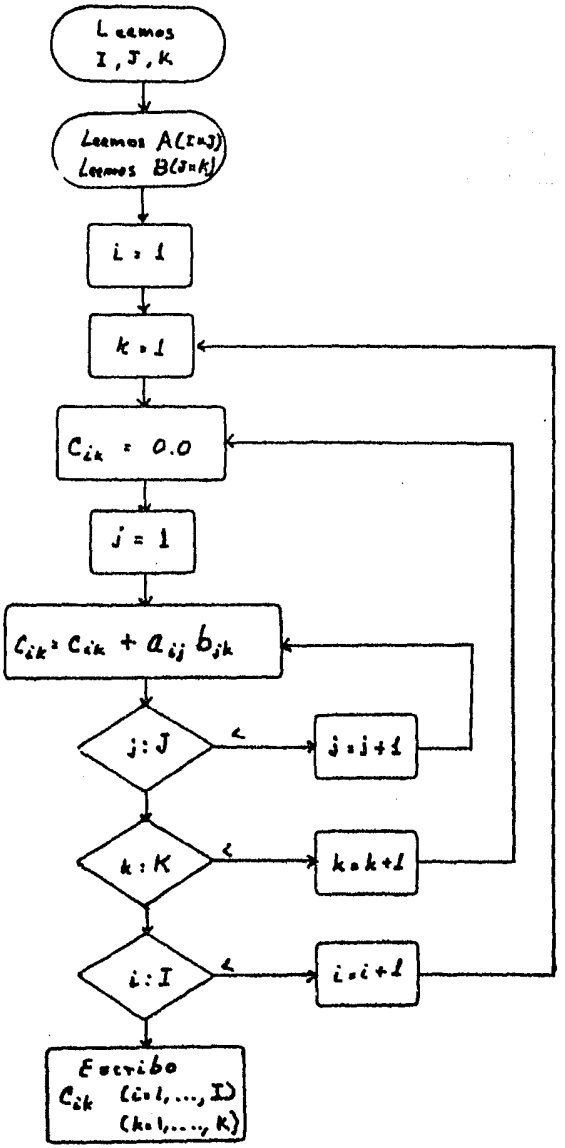

Leemos
I, J, K
Leemos A(I=J)
Leemos B(J=K)
i = 1
k = 1
C_{ik} = 0.0
j = 1
$C_{ik} = C_{ik} + a_{ij} b_{jk}$
j : J
j = j+1
k : K
k = k+1
i : I
i = i+1
Escribo
C_{ik} (i=1,..., I)
(k=1,..., K)

FIG. 4.1

PROGRAMA ESCRITO POR EL USUARIO

```
      PROGRAM MATMUL
C
 ***************************************************************
C     PROGRAMA PARA PRODUCTO DE MATRICES
C     C(I X K) = A(I X J) * B(J X K)
C
 ***************************************************************
C
C
 ***************************************************************
C     PASO 0 : DEFINIMOS LAS MATRICES A, B, C Y LAS VARIABLES IC,
 JC,
C              KC, MC.
C
 ***************************************************************
C
      %DEFMAT(A, 20, 20)
      %DEFMAT(B, 20, 20)
      %DEFMAT(C, 20, 20)
      %LEEI(IC, JC, KC, MC)
C
C
 ***************************************************************
C     PASO 1 : LEEMOS LOS VALORES DE LA MATRIZ A Y DE LA MATRIZ B
C
 ***************************************************************
C
      %ALIMMAT(A, IC, JC)
      %ALIMMAT(B, JC, KC)
C
C
 ***************************************************************
C     PASO 2 : HACEMOS LA OPERACION C(I X K) = A(I X J) * B(J X K)
C
 ***************************************************************
C
      %MULTMAT(A, B, IC, JC, KC)
C
C
 ***************************************************************
C     PASO 3 : ESCRIBIMOS LA MATRIZ C
C
 ***************************************************************
C
      %ESCMAT(C, IC, KC)
       END
.NOLITERAL
```

PROGRAMA RESULTADO DE MPAN

```
      PROGRAM MATMUL
C
 ****************************************************************
C       PROGRAMA PARA PRODUCTO DE MATRICES
C       C(I X K) = A(I X J) * B(J X K)
C
 ****************************************************************
C
C
 ****************************************************************
C       PASO 0 : DEFINIMOS LAS MATRICES A, B, C Y LAS VARIABLES IC,
 JC,
C                KC, MC.
C
 ****************************************************************
C
      DIMENSION A ( 20 , 20 )
      DIMENSION B ( 20 , 20 )
      DIMENSION C ( 20 , 20 )
       READ ( 5 , 101 ) IC , JC , KC , MC
101   FORMAT ( 4I8 )
C
C
 ****************************************************************
C       PASO 1 : LEEMOS LOS VALORES DE LA MATRIZ A Y DE LA MATRIZ B
C
 ****************************************************************
C
         DO 102 MII = 1 , IC
       WRITE ( 5 , 103 ) MII
       READ ( 5 , 104 ) ( A ( MII , MJJ ) , MJJ = 1 , JC )
104   FORMAT ( F10.4 )
102   CONTINUE
103   FORMAT ( ' ALIMENTE USTED LA HILERA ' , I4 )
         DO 105 MII = 1 , JC
       WRITE ( 5 , 106 ) MII
       READ ( 5 , 107 ) ( B ( MII , MJJ ) , MJJ = 1 , KC )
107   FORMAT ( F10.4 )
105   CONTINUE
106   FORMAT ( ' ALIMENTE USTED LA HILERA ' , I4 )
C
C
 ****************************************************************
C       PASO 2 : HACEMOS LA OPERACION C(I X K) = A(I X J) * B(J X K)
C
 ****************************************************************
C
         DO 108 MII = 1 , IC
         DO 109 MKK = 1 , KC
       C ( MII , MKK ) = 0.0
```

```
          DO 110 MJJ = 1 , JC
        C ( MII , MKK ) = C ( MII , MKK ) +
     1 A ( MII , MJJ ) * B ( MJJ , MKK )
110     CONTINUE
109     CONTINUE
108     CONTINUE
C
C
 ***************************************************************
C       PASO 3 : ESCRIBIMOS LA MATRIZ C
C
 ***************************************************************
C
          DO 111 MII = 1 , IC
        WRITE ( 5 , 112 ) MII
       WRITE ( 5 , 113 ) ( C ( MII , MJJ ) , MJJ = 1 , KC )
113     FORMAT ( F10.4 , X )
111     CONTINUE
112     FORMAT ( ' ESCRIBE EL VECTOR ' , I4 )
          END
.NOLITERAL
```

RESULTADOS DEL PROGRAMA ESCRITO

POR EL USUARIO

```
>RUN SAL2
00000003000000030000000300000000
 ALIMENTE USTED LA HILERA     1
1.0
2.0
3.0
 ALIMENTE USTED LA HILERA     2
4.0
5.0
6.0
 ALIMENTE USTED LA HILERA     3
6.0
5.0
4.0
 ALIMENTE USTED LA HILERA     1
3.0
2.0
1.0
 ALIMENTE USTED LA HILERA     2
3.0
4.0
5.0
 ALIMENTE USTED LA HILERA     3
7.0
3.0
1.0
 ESCRIBE EL VECTOR     1
   30.0000
   19.0000
   14.0000
 ESCRIBE EL VECTOR     2
   69.0000
   46.0000
   35.0000
 ESCRIBE EL VECTOR     3
   61.0000
   44.0000
   35.0000
TT6  --  STOP
>
```

4.4. Método para encontrar la inversa de una Matríz por eliminación pivotal

Para encontrar la Matríz inversa de A (A^{-1}) por eliminación pivotal lo hacemos de la siguiente manera:

PASO 0 : Entrada e inicialización. Leemos EPS, N (N = número de renglones (columnas) de A).

Leemos los elementos de la Matríz A (N x N).

Creamos la Matríz idéntica (I) y la llamamos B.

Inicializamos el determinante DEL en 1: DEL = 1.

PASO 1 : Encontramos el elemento pivotal (el elemento de máxima magnitud en la columna K que se encuentre después del elemento de la diagonal). Comparamos $|A_{kk}|$, $|A_{k+1,K}|$, ..., $|A_{nk}|$ para encontrar el elemento mayor, es decir, $|A_{imax,k}|$, intercambiamos el renglón imax de A con el renglón k de A (lo mismo se hace con B).

Si imax $\neq$ k, DEL = -DEL.

El elemento $|A_{kk}|$ es ahora el elemento mayor del conjunto

$$\{|A_{kk}|, |A_{k+1,k}|, \ldots, |A_{nk}|\}.$$

PASO 2 : Preguntamos si el máximo elemento es una aproximación a 0. Si $|A_{kk}| \leq$ EPS, se va a error y termina (es una Matríz singular).

Si $|A_{kk}| >$ EPS, continuamos al paso 3.

PASO 3 : Ejecutamos el K-ésimo estado de reducciones.

a) Renglón k = Renglón k / A_{kk}

DEL = A_{kk} * DEL

Dejamos que DIV = A_{kk}

$A_{kj} = A_{kj}$ / DIV $j = 1, \ldots, N$

$B_{kj} = B_{kj}$ / DIV

b) Renglón i = Renglón i - A_{ik}* Renglón k.

Dejemos que MULT = A_{ik}

$A_{ij} = A_{ij}$ - MULT A_{kj} $i \neq k$, $j = 1, \ldots, N$

$B_{ij} = B_{ij}$ - MULT B_{kj}

PASO 4 : Preguntamos por el contador K. Si K<N, incrementamos K en 1 (K = K + 1) y regresamos al paso 1. Si K>=N, continuamos al paso 5.

PASO 5 : Escribimos la salida A^{-1} = B y |A| = DEL.

Ver (Mc Call)

El diagrama de flujo para encontrar la Matríz inversa de A (A^{-1}) se encontrará en la Fig. 4.2.

Al igual que en la Multiplicación de Matrices al revisar la Biblioteca de MPAN sabremos con qué herramientas contamos para la resolución de este método.

- DEFMAT (W1, W2, W3)

  La cual nos define una Matríz. Entonces para definir la Matríz A de 10 x 10, se sustituye W1 por A, W2 por 10, W3 por 10. Podemos utilizar esta misma Macro para definir la Matríz B de 10 x 10.

- LEEI1 (W1)

  Con la cual podemos leer el valor entero de N si sustitu<u>i</u>mos W1 por N.

- ALIMMAT (W1, W2, W3)

  Con la cual podemos leer una Matríz, si sustituimos W1 por A, W2 por N, W3 por N.

- IDENTI (W1, W2)

  Genera una Matríz idéntica (I) W1 de W2 x W2. Con esta Macro podemos generar la Matríz idéntica I la cual llamaremos B, de N x N, si sustituimos W1 por B y W2 por N.

- MAYDIAG (W1, W2, W3, W4)

  Dada las Matrices W1, W2 cuyas dimensiones son W3 x W4, encontramos el elemento mayor de una columna de la Matríz W1 e intercambiamos los renglones en ambas Matrices. Con esta Macro encontramos el elemento pivotal de la Matríz A e intercambiamos los renglones en ambas Matrices A y B. Si sustituimos W1 por A, W2 por B, W3 por N y W4 por K.

- DIVPIVOT (W1, W2, W3, W4, W5)

  Dadas las Matrices W1 y W2, se realizan las siguientes operaciones en ambas Matrices:

  Renglón W4 = Renglón W4 / $A_{W4,W4}$

  W3 = $A_{W4,W4}$*W3

  Donde W5 nos indica de 1 a qué varía el renglón de la Matríz Si sustituimos W1 por A, W2 por B, W3 por DEL, W4 por K y W5 por N obtenemos el inciso a) del paso 3.

- CONLINPIV (W1, W2, W3, W4)

  Dadas las Matrices W1 y W2, se realizó la siguiente operación en ambas Matrices.

  Renglón W3 = Renglón W3 - $A_{W3,W4}$* Renglón W4

  Si sustituimos W1 por A, W2 por B, W3 por K, W4 por N obtendremos el inciso b) del paso 3.

- ESCMAT (W1, W2, W3)

  Esta Macro escribe o da los resultados de una Matríz W1 de W2 x W3. Si queremos conocer cuál es la inversa de la Matríz A (A^{-1} = B), escribimos la Matríz B sustituyendo W1 por B, W2 por N, W3 por N.

- ESCF (W1)

  Con esta Macro escribimos el valor real de W1. Otro dato importante de conocer es el valor del determinante, si sustituimos W1 por DEL obtenemos ese valor.

Con la utilización de estos Macros y otros Macros que se encuentran en la primera y segunda capa resolveremos el Método para encontrar la inversa de una Matríz por eliminación pivotal. A continuación se mostrará el programa documentado (antes de entrar a MPAN), el programa en Fortran (resultado de MPAN), el cual se compila, liga y ejecuta de la siguiente manera:

```
FOR SALIDA = SALIDA
TKB SALIDA = SALIDA
TKB SALIDA = SALIDA
RUN SALIDA
```

Obteniendo los resultados que también se muestran a continuación:

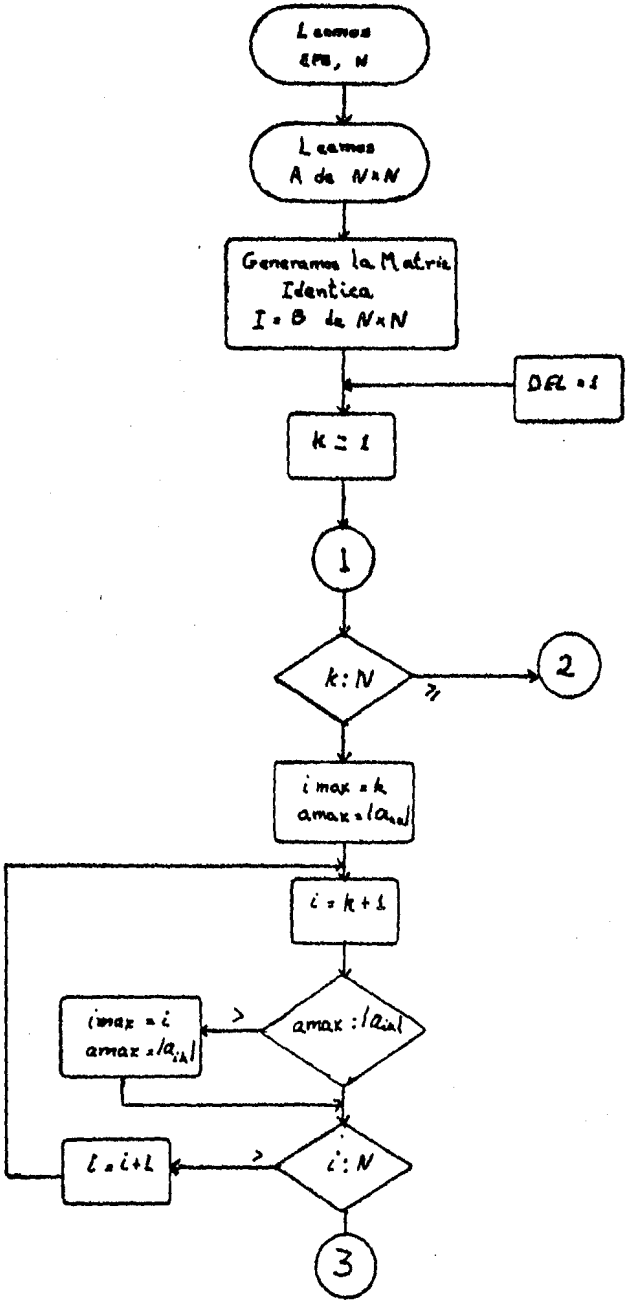
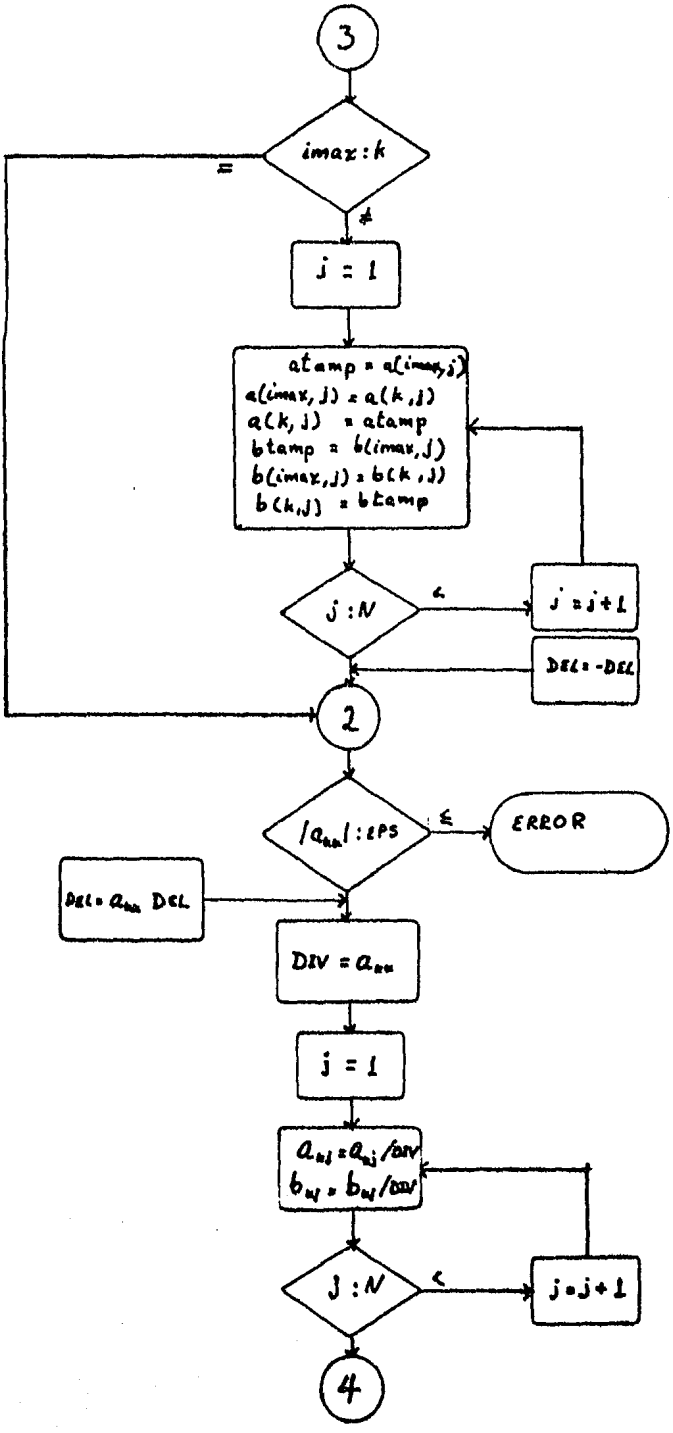
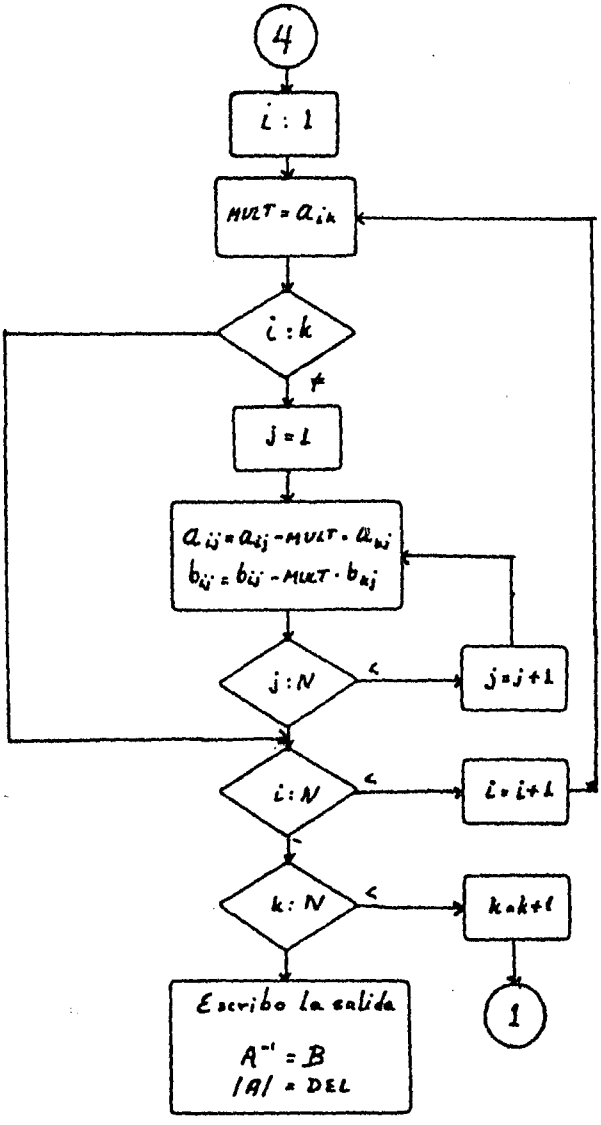

Leemos
EPS, N
Leemos
A de N×N
Generamos la Matriz
Identica
I = B de N×N
DAL = 1
K = 1
1
k : N
2
imax = k
amax = |a_kk|
i = k+1
imax = i
amax = |a_ik|
amax : |a_ik|
i = i+1
i : N
3

FIG. 4.2


3
imax : k
=
≠
j = 1
atamp = a(imax,j)
a(imax,j) = a(k,j)
a(k,j) = atamp
btamp = b(imax,j)
b(imax,j) = b(k,j)
b(k,j) = btamp
j : N
<
j = j+1
DEL = -DEL
2
|a_kk| : EPS
≤
ERROR
DEL = a_kk DEL
DIV = a_kk
j = 1
a_kj = a_kj / DIV
b_kj = b_kj / DIV
j : N
<
j = j+1
4

Cont. Fig. 4.2


4
i : 1
MULT = a_ik
i : k
≠
j = 1
a_ij = a_ij - MULT · a_kj
b_ij = b_ij - MULT · b_kj
j : N
<
j = j+1
i : N
<
i = i+1
k : N
<
k = k+1
1
Escribo la salida
A^-1 = B
|A| = DEL

PROGRAMA ESCRITO POR EL USUARIO

```
      PROGRAM INVERSE
C
 ********************************************************************
C     ENCONTRAR LA MATRIZ INVERSA DE A CON PIVOTEO PARCIAL.
C     A = Matriz original      B = Matriz inversa
C
 ********************************************************************
C
      %DEFMAT(A, 20, 20)
      %DEFMAT(B, 20, 20)
C
C
 ********************************************************************
C     PASO 0 : ENTRADA E INICIALIZACION
C
 ********************************************************************
C
      %LEEI1(N)
      %LEEF(EPS)
      %ALIMMAT(A, N, N)
      %IDENTI(B, N)
      DEL = 1.0
C
C
 ********************************************************************
C     PASO 1 : ENCUENTRA EL ELEMENTO PIVOTAL
C
 ********************************************************************
C
      K = 1
      %WHILE(CK .LE. NJ)
      %MAYDIAG(A, B, N, K)
C
C
 ********************************************************************
C     PASO 2 : PREGUNTA SI EL ELEMENTO MAXIMO ES 0, ES DECIR,
C              A(K,K) <= EPS TERMINO, SI NO ME VOY AL PASO 3.
C
 ********************************************************************
C
       %GENETI()
       %PUSH
       IF((ABS(A(K,K)) - EPS) .LE. 0) GO TO %ESC
C
C
 ********************************************************************
C     PASO 3 : REALIZA EL PROCESO DE REDUCCION DE K ESTADOS
C
 ********************************************************************
C
```

```
         %DIVPIVOT(A, B, DEL, K, N)
        %CONLINPIV(A, B, K, N)
C
C
 ***************************************************************
C       PASO 4 : DEPENDIENDO DE K. SI K > N CONTINUAMOS AL PASO 5,
 SINO
C                REGRESAMOS AL PASO 1.
C
 ***************************************************************
C
        %AUX1ETI()
        %POP
        %ETIAUX()
        K = K + 1
        %ENDW()
C
 ***************************************************************
C       PASO 5 : ESCRIBIMOS LA SALIDA     A-1 = B
C                                         |A| = DEL
C
 ***************************************************************
C
       %PUSH
       %ETIAUX1()
       %ESCLET([B = INVERSA])
       %ESCMAT(B, N, N)
       %ESCLET([DEL = DETERMINANTE])
       %ESCF(DEL)
       %POP
%ESC   STOP
       END
.NOLITERAL
```

PROGRAMA RESULTADO DE MPAN

```
      PROGRAM INVERSE
C
C*****************************************************************
C       ENCONTRAR LA MATRIZ INVERSA DE A CON PIVOTEO PARCIAL.
C       A = Matriz original      B = Matriz inversa
C
C*****************************************************************
C
      DIMENSION A ( 20 , 20 )
      DIMENSION B ( 20 , 20 )
C
C
C*****************************************************************
C       PASO 0 : ENTRADA E INICIALIZACION
C
C*****************************************************************
C
       READ ( 5 , 101 ) N
101   FORMAT ( I8 )
      READ ( 5 , 102 ) EPS
102   FORMAT ( F10.8 )
         DO 103 MII = 1 , N
       WRITE ( 5 , 104 ) MII
       READ ( 5 , 105 ) ( A ( MII , MJJ ) , MJJ = 1 , N )
105   FORMAT ( F10.4 )
103   CONTINUE
104   FORMAT ( ' ALIMENTE USTED LA HILERA ' , I4 )
         DO 106 MII = 1 , N
         DO 107 MJJ = 1 , N
       IF ( ( MII - MJJ ) .EQ. 0 ) GO TO 108
         B ( MII , MJJ ) = 0.0
         GO TO 107
108   B ( MII , MJJ ) = 1.0
107   CONTINUE
106   CONTINUE
       DEL = 1.0
C
C
C*****************************************************************
C       PASO 1 : ENCUENTRA EL ELEMENTO PIVOTAL
C
C*****************************************************************
C
       K = 1
109   CONTINUE
         IF ( .NOT. ( K .LE. N  ) ) GO TO 110
       IF ( ( K - N ) .GE. 0 ) GO TO 111
          IMAX = K
          AMAX = ABS ( A ( K , K ) )
          KP1 = K + 1
```

```
          DO 112 MII = KP1 , N
         IF ( ( AMAX - ABS ( A ( MII , K ) ) ) .GE. 0 ) GO TO 112
          IMAX = MII
          AMAX = ABS ( A ( MII , K ) )
112     CONTINUE
         IF ( ( IMAX - K ) .EQ. 0 ) GO TO 111
          DO 113 MJJ = 1 , N
           ATMP = A ( IMAX , MJJ )
           A ( IMAX , MJJ ) = A ( K , MJJ )
           A ( K , MJJ ) = ATMP
113     CONTINUE
          DO 114 MJJ = 1 , N
           ATMP = B ( IMAX , MJJ )
           B ( IMAX , MJJ ) = B ( K , MJJ )
           B ( K , MJJ ) = ATMP
114     CONTINUE
         DEL = - DEL
111     CONTINUE
C
C
 ***********************************************************************
C       PASO 2 : PREGUNTA SI EL ELEMENTO MAXIMO ES 0, ES DECIR,
C                A(K,K) <= EPS TERMINO, SI NO ME VOY AL PASO 3.
C
 ***********************************************************************
C
         IF ( ( ABS ( A ( K , K ) ) - EPS ) .LE. 0 ) GO TO 115
C
C
 ***********************************************************************
C         PASO 3 : REALIZA EL PROCESO DE REDUCCION DE K ESTADOS
C
 ***********************************************************************
C
         DEL = A ( K , K ) * DEL
         DIV = A ( K , K )
          DO 116 MJJ = 1 , N
           A ( K , MJJ ) = A ( K , MJJ ) / DIV
116     CONTINUE
          DO 117 MJJ = 1 , N
           B ( K , MJJ ) = B ( K , MJJ ) / DIV
117     CONTINUE
          DO 118 MII = 1 , N
           AMULT = A ( MII , K )
           IF ( ( MII - K ) .EQ. 0 ) GO TO 118
          DO 119 MJJ = 1 , N
         A ( MII , MJJ ) = A ( MII , MJJ ) - AMULT * A ( K , MJJ )
119     CONTINUE
          DO 120 MJJ = 1 , N
         B ( MII , MJJ ) = B ( MII , MJJ ) - AMULT * B ( K , MJJ )
120     CONTINUE
118     CONTINUE
C
```

```
C
 ***************************************************************
C       PASO 4 : DEPENDIENDO DE K. SI K > N CONTINUAMOS AL PASO 5,
 SINO
C                REGRESAMOS AL PASO 1.
C
 ***************************************************************
C
        K = K + 1
          GO TO 109
110     CONTINUE
C
 ***************************************************************
C       PASO 5 : ESCRIBIMOS LA SALISA     A-1 = B
C                                         !A! = DEL
C
 ***************************************************************
C
       WRITE ( 5 , 121 )
121    FORMAT ( ' B = INVERSA  ' )
          DO 122 MII = 1 , N
        WRITE ( 5 , 123 ) MII
       WRITE ( 5 , 124 ) ( B ( MII , MJJ ) , MJJ = 1 , N )
124    FORMAT ( F10.4 , X )
122    CONTINUE
123    FORMAT ( ' ESCRIBE EL VECTOR ' , I4 )
       WRITE ( 5 , 125 )
125    FORMAT ( ' DEL = DETERMINANTE  ' )
       WRITE ( 5 , 126 ) DEL
126    FORMAT ( F10.4 )
115    STOP
       END
.NOLITERAL
```

RESULTADOS DEL PROGRAMA ESCRITO

POR EL USUARIO

```
 RUN SAL4
00000004
0.9
 ALIMENTE USTED LA HILERA     1
4.0
8.0
2.0
1.0
 ALIMENTE USTED LA HILERA     2
1.0
5.0
3.0
8.0
 ALIMENTE USTED LA HILERA     3
2.0
7.0
1.0
4.0
 ALIMENTE USTED LA HILERA     4
3.0
8.0
2.0
1.09
 B = INVERSA
 ESCRIBE EL VECTOR     1
    1.0148
    0.0063
    0.0148
   -1.0317
 ESCRIBE EL VECTOR     2
   -0.3581
   -0.0821
    0.1419
    0.4103
 ESCRIBE EL VECTOR     3
   -0.1790
    0.2804
   -0.6790
    0.5979
 ESCRIBE EL VECTOR     4
    0.1641
    0.0703
    0.1641
   -0.3517
 DEL = DETERMINANTE
   85.3000
TT6  --  STOP
>
```

4.5. Resolución AX = B por el Método GAUSS-SEIDEL

Para resolver la ecuación AX = B por el Método iterativo de GAUSS-SEIDEL, se hace de la siguiente manera:

PASO 0 : Entrada de parámetros y datos.

N = Número de ecuaciones

KC = Máximo número de iteraciones

EPS = Criterio de convergencia

La Matríz argumento está formada por A,B de N x N+1, tal que: $A_{i,n+1} = B_i$ $(i = 1,\ldots, N)$

PASO 1 : Divide la i-ésima ecuación por el elemento de la diagonal A_{ii} $(A_{ii} \neq 0)$.

$$A_{ij} = A_{ij} / A_{ii} \quad (i = 1,\ldots, N \; ; \; J = 1,\ldots,N+1)$$

PASO 2 : Inicializa el contador de las iteraciones K (K = 1). Inicializa o limpia con 0's los componentes del vector X_i^1 donde

$$X_i^1 = 0.0 \qquad i = 1,\ldots, N$$

PASO 3 : Calcula las sucesivas iteraciones X_i^{k+1} usando la siguiente fórmula:

$$X_i^{k+1} = A_{i,n+1} - \sum_{j=1}^{i-1} A_{ij} A_j^{k+1} - \sum_{j=i+1}^{N} A_{ij} X_j^k \quad i = 1,\ldots,N$$

PASO 4 : Pregunta por la convergencia:

a) Si algún $|X_i^{k+1} - X_i^k| > EPS$ ves al paso 5.

b) Si para todo $|X_i^{k+1} - X_i^k| <= EPS$ ves al paso 6.

PASO 5 : Pregunta por el contador de iteraciones

a) Si K< KC incrementa K en 1 (K = K + 1) y regresa al paso 3.

b) Si K>=KC ve al paso 7.

PASO 6 : Si la convergencia fue un éxito. Escribe la solución:

$X_i = X_i^{k+1} \qquad i = 1, \ldots, N$

PASO 7 : Si no escribe "El programa falló en la KCésima iteración". (ver Mc Call).

El diagrama de flujo para encontrar la solución de la ecuación AX = B, por el método iterativo de GAUSS-SEIDEL se encontrará en la Fig. 4.3.

Al igual que en los dos problemas anteriores en la Biblioteca de MPAN encontraremos las herramientas necesarias, para la resolución de este método.

- DEFMAT (W1, W2, W3)

  La cual nos define una Matríz. Entonces para definir la Matríz A de 20 x 21, se sustituye W1 por A, W2 por 20 y W3 por 21.

- DEFVEC (W1, W2)

  La cual define un vector. Entonces para definir el vector XK de 20, se sustituye W1 por XK y W2 por 20. Podemos utilizar esta misma Macro para definir el vector XKP1 (20).

- LEEI (W1, W2, W3, W4)

  Con el cual podemos leer los valores enteros de N, NP1,

KC, N1 si sustituimos W1 por N, W2 por NP1, W3 por KC y W4 por N1.

- LEEF (W1)

Con el cual podemos leer el valor real de EPS, si sustituimos W1 por EPS.

- ALIMMAT (W1, W2, W3)

Con la cual podemos leer una Matríz si sustituimos W1 por A, W2 por N y W3 por NP1.

- DIVDIAG (W1, W2, W3)

Divide la i-ésima ecuación de la Matríz W1 por el elemento de su diagonal A_{ii} ($A_{ii} \neq 0$), donde

$$A_{ij} = A_{ij} / A_{ii} \quad i = 1,\ldots, W2 \; ; \; j = 1, \ldots, W3)$$

Si sustituimos W1 por A, W2 por N y W3 por NP1 obtenemos el paso 1 del problema.

- INICCONS (W1, W2)

Le asignamos el valor numérico W2 a la variable W1. Si se quiere inicializar K en 1 (K = 1) se puede hacer sustituyendo W1 por K y W2 por 1.

- LIMVEC (W1, W2)

Inicializa o limpia con 0's el vector W1 (W2). Si se quiere limpiar el vector XK (N) se sustituye W1 por XK y W2 por N.

- ITERA (W1, W2, W3, W4, W5)

Dada la Matríz W1 y los vectores W2 y W3 se calculan las sucesivas iteraciones X_i^{k+1} usando la siguiente fórmula:

$$x_i^{k+1} = A_{i,n+1} - \sum_{j=1}^{i-1} A_{ij}x_j^{k+1} - \sum_{j=i+1}^{W5} A_{ij}x_j^{k} \quad i = 1,\ldots, W4$$

si sustituimos W1 por A, W2 por XKP1, W3 por XK, W4 por N y W5 por NP1 obtenemos el paso 3 del problema.

- ESCV (W1, W2)

  Este Macro escribe o da los resultados del vector W1 (W2). Si queremos conocer el vector solución, se debe sustituir W1 por XKP1 y W2 por N (esto siempre y cuando la convergencia exista).

- ESCI (W1)

  Esta Macro escribe el valor entero de W1. Si la convergencia falló es necesario saber en qué k-ésima iteración se quedó, para eso sustituimos W1 por KC.

Con la utilización de este y otros Macros que se encuentran en la primera y segunda capa resolveremos el Método para encontrar la solución a la ecuación AX = B por el método iterativo de GAUSS-SEIDEL, ayudándonos de instrucciones del lenguaje base así como de la función iterativa REPEAT-UNTIL.

A continuación se mostrará el programa documentado (antes de entrar a MPAN), el programa en Fortran resultado de MPAN, el cual se compila, liga y ejecuta de la siguiente manera:

```
FOR SALIDA = SALIDA
TKB SALIDA = SALIDA
RUN SALIDA
```

Obteniendo los resultados que también se muestran a continuación.

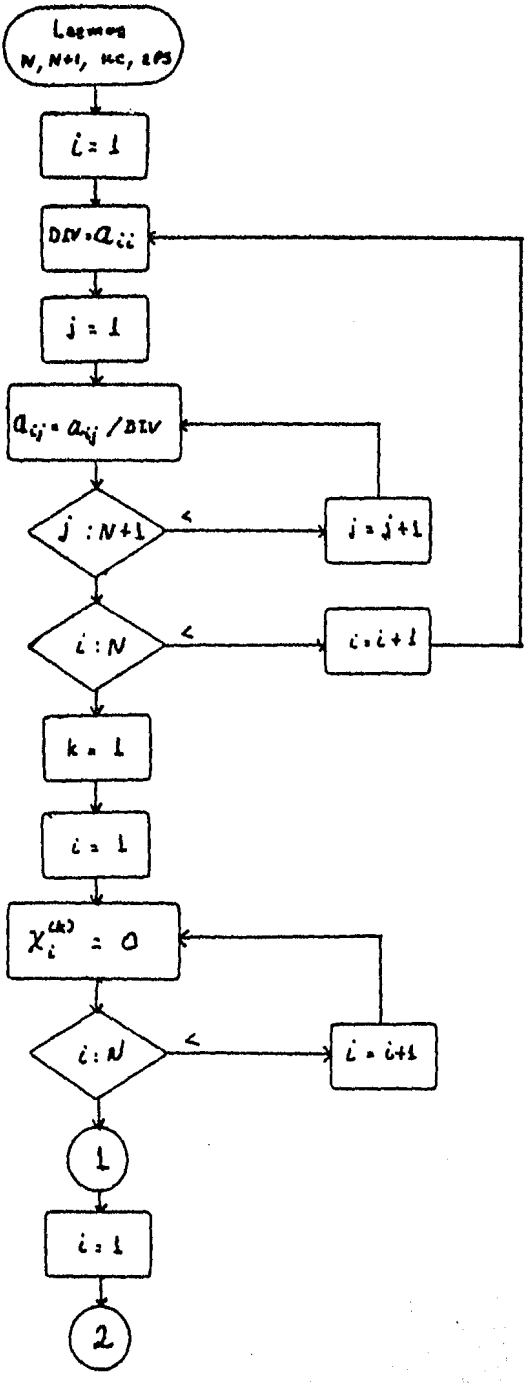
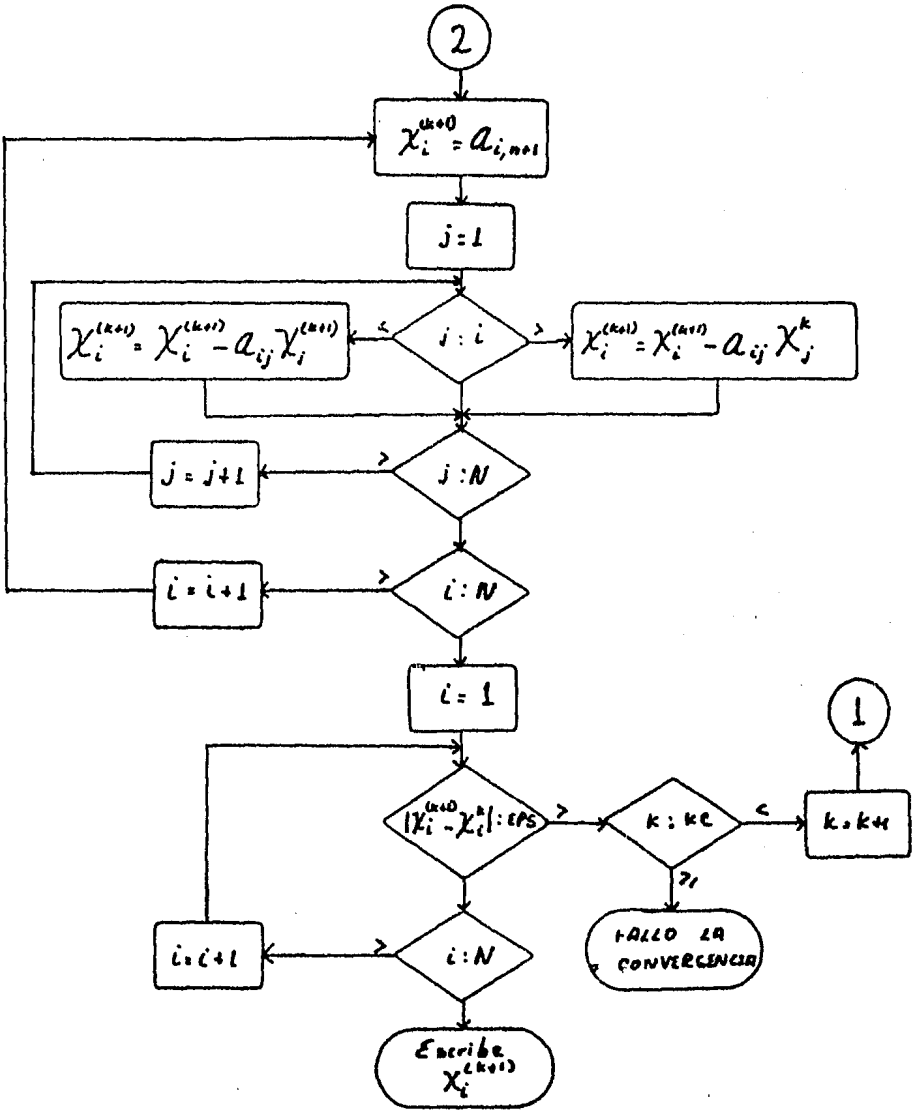

Lesmes
N, N+1, MC, EPS
i = 1
DIV = a_ii
j = 1
a_ij = a_ij / DIV
j : N+1
<
j = j+1
i : N
<
i = i+1
k = 1
i = 1
χ_i^(k) = 0
i : N
<
i = i+1
1
i = 1
2

FIG. 4.3


2
$X_i^{(k+1)} = a_{i,n+1}$
j = 1
$X_i^{(k+1)} = X_i^{(k+1)} - a_{ij} X_i^{(k+1)}$
<
j : i
>
$X_i^{(k+1)} = X_i^{(k+1)} - a_{ij} X_j^k$
j = j+1
>
j : N
i = i+1
>
i : N
i = 1
1
$|X_i^{(k+1)} - X_i^k|$: EPS
>
k : kc
<
k = k+1
>1
HALLO LA CONVERGENCIA
i = i+1
>
i : N
Escribe $X_i^{(k+1)}$

PROGRAMA ESCRITO POR EL USUARIO

```
      PROGRAM GAUSSSEIDEL
C
 ***************************************************************
C     RESUELVE AX = B POR EL METODO ITERATIVO GAUSS-SEIDEL. DONDE
C     LA MATRIZ ARGUMENTO LA COMPONEN A Y B.
C
 ***************************************************************
C
      %MACRO ESCCONV()
      %ESCLET(LA SOL DE X(I) =)
      %ESCV(XKP1, N)
      %GENETI()
      %PUSH
%ESC  STOP
      %ENDM
      %MACRO ESCFALLO()
      %ESCLET( FALLO EN KC = )
      %ESCI(KC)
      %POP()
       GO TO %ESC
      %ENDM
      %DEFMAT(A, 20, 21)
      %DEFVEC(XK, 20)
      %DEFVEC(XKP1, 20)
C
C
 ***************************************************************
C     PASO 0 : PARAMETROS Y DATOS DE ENTRADA
C
 ***************************************************************
C
      %LEEI(N, NP1, KC, N1)
      %LEEF(EPS)
      %ALIMMAT(A, N, NP1)
C
C
 ***************************************************************
C     PASO 1 : DIVIDE CADA RENGLON POR EL ELEMENTO DE SU DIAGONAL.
C
 ***************************************************************
C
      %DIVDIAG(A, N, NP1)
C
C
 ***************************************************************
C     PASO 2 : INICIALIZA K Y EL VECTOR X(I)
C
 ***************************************************************
C
      %INICCONS(K, 1)
```

```
      %LIMVEC(XK, N)
C
C
C*****************************************************************
C     PASO 3 : CALCULA LAS SUCESIVAS ITERACIONES DE Xi, k+1
C
C*****************************************************************
C
10    CONTINUE
      %ITERA(A, XKP1, XK, N, NP1)
C
C
C*****************************************************************
C     PASO 4 : PREGUNTA POR LA CONVERGENCIA, QUE DEPENDE DE EPS,
C              SI CONVERGE ==> PASO 6,  SI NO ==> PASO 5
C
C*****************************************************************
C
       %INICIND(I)
       %REPEAT()
        I = I + 1
        IF((ABS(XKP1(I) - XK(I)) - IEPS) .GT. 0)
     1   GO TO 50
       %UNTIL(I .GT. N)
C
C
C*****************************************************************
C     PASO 6 : SE ESCRIBE Xi PORQUE CONVERGE Y TERMINA
C
C*****************************************************************
C
      %ESCCONV()
C
C
C*****************************************************************
C     PASO 5 : PREGUNTA POR EL CONTADOR DE LA ITERACION
C              SI K < KC      K = K + 1 ==> PASO 3
C              SI K >= KC               ==> PASO 7
C
C*****************************************************************
C
50    CONTINUE
       IF(K - KC) 51, 55, 55
51    CONTINUE
         %CONVERGE(XKP1, XK, N, K)
       GO TO 10
C
C
C*****************************************************************
C     PASO 7 : ESCRIBE 'EL PROGRAMA FALLO EN KC ='
C
C*****************************************************************
C
```

```
55          CONTINUE
           %ESCFALLO()
           END
.NOLITERAL
```

PROGRAMA RESULTADO DE MPAN

```
      PROGRAM GAUSSSEIDEL
C
C*****************************************************************
C       RESUELVE AX = B POR EL METODO ITERATIVO GAUSS-SEIDEL. DONDE
C       LA MATRIZ-ARGUMENTO LA COMPONEN A Y B.
C
C*****************************************************************
C
      DIMENSION A ( 20 , 21 )
      DIMENSION XK ( 20 )
      DIMENSION XKP1 ( 20 )
C
C
C*****************************************************************
C       PASO 0 : PARAMETROS Y DATOS DE ENTRADA
C
C*****************************************************************
C
       READ ( 5 , 101 ) N , NP1 , KC , N1
101   FORMAT ( 4I8 )
      READ ( 5 , 102 ) EPS
102   FORMAT ( F10.8 )
         DO 103 MII = 1 , N
       WRITE ( 5 , 104 ) MII
       READ ( 5 , 105 ) ( A ( MII , MJJ ) , MJJ = 1 , NP1 )
105   FORMAT ( F10.4 )
103   CONTINUE
104   FORMAT ( ' ALIMENTE USTED LA HILERA ' , I4 )
C
C
C*****************************************************************
C       PASO 1 : DIVIDE CADA RENGLON POR EL ELEMENTO DE SU DIAGONAL.
C
C*****************************************************************
C
         DO 106 MII = 1 , N
       DIV = A ( MII , MII )
         DO 107 MJJ = 1 , NP1
          A ( MII , MJJ ) = A ( MII , MJJ ) / DIV
107   CONTINUE
106   CONTINUE
C
C
C*****************************************************************
C       PASO 2 : INICIALIZA K Y EL VECTOR X(I)
C
C*****************************************************************
C
        K = 1
         DO 108 MII = 1 , N
```

```
        XK ( MII ) = 0.0
108     CONTINUE
C
C
 ****************************************************************
C        PASO 3 : CALCULA LAS SUCESIVAS ITERACIONES DE Xi, k+1
C
 ****************************************************************
C
10      CONTINUE
           DO 109 MII = 1 , N
        XKP1 ( MII ) = A ( MII , NP1 )
           DO 110 MJJ = 1 , N
        IF ( ( MJJ - MII ) .EQ. 0 ) GO TO 110
        IF ( ( MJJ - MII ) .GT. 0 ) GO TO 111
          XKP1 ( MII ) = XKP1 ( MII ) - A ( MII , MJJ ) * XKP1 ( MJJ )
        GO TO 110
111     XKP1 ( MII ) = XKP1 ( MII ) - A ( MII , MJJ ) * XK ( MJJ )
110     CONTINUE
109     CONTINUE
C
C
 ****************************************************************
C        PASO 4 : PREGUNTA POR LA CONVERGENCIA, QUE DEPENDE DE EPS,
C                 SI CONVERGE ==> PASO 6,  SI NO ==> PASO 5
C
 ****************************************************************
C
          I = 0
112     CONTINUE
           I = I + 1
           IF ( ( ABS ( XKP1 ( I ) - XK ( I ) ) - IEPS ) .GT. 0 )
     1 GO TO 50
           IF ( .NOT. ( I .GT. N ) ) GO TO 112
C
C
 ****************************************************************
C         PASO 6 : SE ESCRIBE Xi PORQUE CONVERGE Y TERMINA
C
 ****************************************************************
C
        WRITE ( 5 , 113 )
113     FORMAT ( ' LA SOL DE X ( I ) = ' )
         WRITE ( 5 , 114 ) ( XKP1 ( MXX ) , MXX = 1 , N )
114     FORMAT ( F10.4 , X )
115     STOP
C
C
 ****************************************************************
C         PASO 5 : PREGUNTA POR EL CONTADOR DE LA ITERACION
C                  SI K < KC     K = K + 1 ==> PASO 3
C                  SI K >= KC              ==> PASO 7
C
```

```
 ***************************************************************
C
50      CONTINUE
           IF ( K - KC ) 51 , 55 , 55
51      CONTINUE
          K = K + 1
           DO 116 I = 1 , N
          XK ( I ) = XKP1 ( I )
116     CONTINUE
           GO TO 10
C
C
 ***************************************************************
C         PASO 7 : ESCRIBE 'EL PROIRAMA FALLO EN KC ='
C
 ***************************************************************
C
55      CONT NUE
        WRITE ( 5 , 117 )
117     FORMAT ( ' FALLO EN KC = I ' )
        WRITE ( 5 , 118 ) KC
118     FORMAT ( I8 )
          GO TO 115
        END
.NOLITERAL
```

RESULTADOS DEL PROGRAMA ESCRITO

POR EL USUARIO

```
>RUN SAL5
00000003000000040000005000000000
0.9
 ALIMENTE USTED LA HILERA     1
4.0
1.0
2.0
16.0
 ALIMENTE USTED LA HILERA     2
1.0
3.0
1.0
10.0
 ALIMENTE USTED LA HILERA     3
1.0
2.0
5.0
12.0
 LA SOL DE X ( I ) =
    3.0000
    2.0000
    1.0000
TT6  --  STOP
>
```

4.6. Conclusiones

Este capítulo demuestra lo útil que puede ser MPAN para resolver problemas de Análisis Numérico.

Con el primer ejemplo, se ve lo simple que puede quedar un programa si la Biblioteca da las herramientas necesarias, para resolver el problema.

Con el segundo ejemplo se ve la utilización de la función iterativa WHILE-ENDW, para controlar el flujo del programa.

Se aprovechó el tercer ejemplo para definir Macros, utilizar la función iterativa REPEAT-UNTIL e incluye instrucciones del lenguaje base.

V. REALIZACION

5.1. Introducción

El Departamento de Matemáticas de la Facultad de Ciencias de la U.N.A.M., cuenta actualmente (1985) con el siguiente equipo de cómputo en sus laboratorios de computación.

a) 3 terminales conectadas a la computadora B-6700

b) 1 minicomputadora PDP-11/34

c) 1 minicomputadora PDP-11/03

d) 1 microcomputadora motorola-6800

e) 1 microcomputadora INTEL-8080

La disponibilidad del equipo se hizo de la siguiente manera:

a) Existen tres terminales conectadas a la B-6700 y la mayoría de los profesores de tiempo completo trabajan solamente en dicha computadora, además de que no todos los estudiantes tienen acceso a dicho equipo.

b) Las microcomputadoras (motorola-6800 y LINTEL-8080) y la minicomputadora PDP-11/03 tienen sistemas para un usuario únicamente, además de contar con Fortran como lenguaje de alto nivel.

c) La PDP-11-34, es la única de todas las computadoras (excluyendo obviamente a la B-6700) que tiene sistema operativo

multi-usuario, además de un equipo periférico que da facilidad de uso en cualquier momento. La PDP-11/34 cuenta con dos unidades de disco, seis terminales y una impresora.

Esta información dada a grandes rasgos, muestra de una manera superficial el porque se escogió a la minicomputadora PDP-11/34 para hacer el presente trabajo.

La minicomputadora PDP-11/34 tiene los siguientes compiladores:

a) FORTRAN

b) PASCAL

c) 'C'

Se optó por el lenguaje 'C', aunque 'C', no es un lenguaje de alto nivel, tampoco se le puede catalogar como un lenguaje de bajo nivel como lo son los lenguajes ensambladores, pero toma características importantes de ambos.

Una de las características importantes de los lenguajes de bajo nivel es el poder manejar: direcciones (directas o indirectas), caracteres o cadena de caracteres y números. Y de los lenguajes de alto nivel utiliza funciones iterativas como son if, while, for, do y switch, a la vez de ser un lenguaje recursivo, ver (Kernighan-2).

Por todas estas características se escribió MPAN en lenguaje 'C'.

Fortran es un lenguaje que no fue construido para el manejo de texto, por lo tanto, su uso para la realización de MPAN tiende a ser difícil y obscura. Más aún, recordando que MPAN admite recursión, es evidente que la mejor manera de manejar problemas recursivos es con un lenguaje recursivo.

Dadas las anteriores argumentaciones, podría pensarse en el compilador PASCAL, ya que este lenguaje cumple con los requisitos anteriormente mencionados. Se pensó en el lenguaje 'C' más que en PASCAL por ser un lenguaje más orientado a la programación de sistemas que PASCAL.

5.2. Descripción del programa

Descripción general

Teniendo determinado el equipo y el lenguaje a usar en la realización de MPAN, debemos pasar a tomar en cuenta algunas consideraciones importantes ya que de éstas dependerá directamente la estructura de MPAN.

a) Debido a que MPAN trabaja en modo de advertencia hay que pensar en un conjunto de caracteres de señal que no sean muy utilizados, por esta razón se eligieron los siguientes caracteres "%" y "∿".

b) Se construyó un analizador léxico de tal manera que con cada llamada a dicho módulo nos entrega un átomo. Este Analizador Léxico evidentemente será la parte central de MPAN ya que todos los demás módulos que necesiten obtener algún símbolo del texto de entrada deberán hacerlo por medio de este Analizador Léxico.
Esto tiene la gran ventaja de que los identificadores están determinados por el Analizador, y si en algún momento se desea que los identificadores sean de otro tipo, será posible con modificar el Analizador únicamente.

c) MPAN tiene dos módulos importantes además del Analizador Léxico, uno de los módulos en el que maneja las Macro-definiciones y el otro módulo es el que maneja las Macro-llamadas.

El módulo que maneja las Macro-definiciones resuelve el problema de las anidaciones por medio de un contador de niveles. Y el módulo de las Macro-llamadas es el que maneja diferentes módulos.dependiendo de lo que encuentre dentro del cuerpo del Macro-llamado.

d) Otra de las características importantes de MPAN es la de manejar Metavariables, las Metavariables las podemos encontrar en el texto fuente (de entrada) y en los cuerpos de los Macros. A las Metavariables se les puede asignar un valor constante o una expresión que contenga Metavariables o constantes.

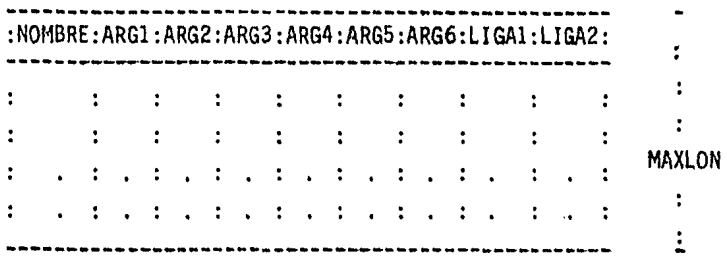
Descripción de tablas

MPAN maneja la mayor parte de su información por medio de las siguientes tablas:

- TABMAC

Esta tabla está compuesta por los nombres de los Macros, los argumentos de los Macros, la dirección de los Macros y la longitud de los Macros. Estos nombres, argumentos, direcciones y longitudes de los Macros son los que se encuentran en la Biblioteca de Macros y los Macros definidos por el usuario en su programa fuente.

Esta tabla tiene la siguiente forma:



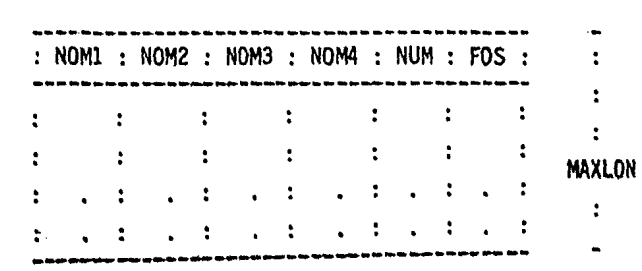
es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

- El nombre del Macro (NOMBRE).
- Los argumentos del Macro (ARG1, ARG2, ARG3, ARG4, ARG5, ARG6).
- El apuntador al archivo auxiliar que contiene al cuerpo del Macro asociado a dicho nombre (LIGA1).
- La longitud de dicho cuerpo del Macro (LIGA2).

- TABSUB

Esta tabla está compuesta por los nombres de las subrrutinas o funciones, el nombre de las subrrutinas o funciones llamada por esa subrrutina o función y la dirección de la subrrutina o función.

Esta tabla tiene la siguiente forma:

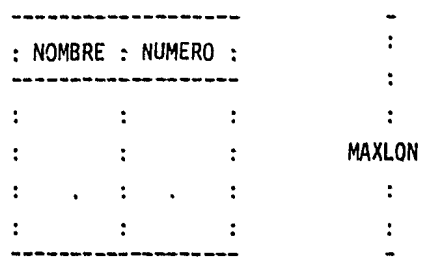


es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

- El nombre de la subrrutina o función llamada (NOM1).
- Los nombres de las subrrutinas o funciones que se encuentran en el cuerpo de la subrrutina o función llamada (NOM2, NOM3, NOM4).
- La dirección de la subrrutina llamada (NUM).
- El valor numérico (1 ó 0) que nos indica si se trata de una subrrutina o función respectivamente (FOS).
- TABMET

  Esta tabla está compuesta por los nombres de las Metavariables y su valor numérico. Estos nombres y valores de las Metavariables son los utilizados en los cuerpos de los Macros y los definidos por el usuario en su programa fuente.

  Esta tabla tiene la siguiente forma:



  es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

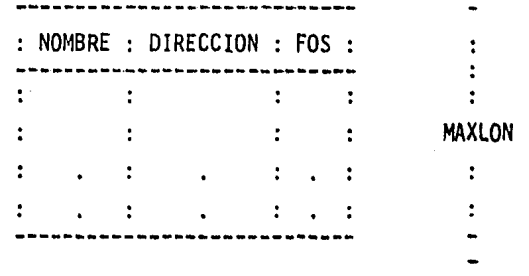
- El nombre de la Metavariable llamada (NOMBRE).

- El valor numérico de la Metavariable llamado (NUMERO).

- TABNOM

Esta tabla está compuesta por los nombres de las subrrutinas o funciones llamadas por el usuario en su programa fuente y su dirección en el archivo de la Biblioteca de Análisis Numérico.

Esta tabla tiene la siguiente forma:



Es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

- El nombre de la subrrutina o función llamada por el usuario en su programa fuente (NOMBRE).

- La dirección de la subrrutina en la Biblioteca de Análisis Numérico (DIRECCION).

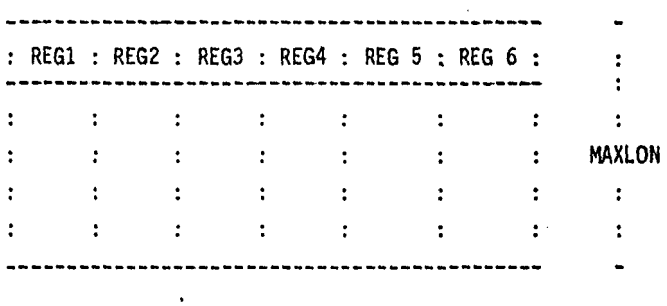
- El valor numérico (1 ó 0) que nos indica si se trata de una subrrutina o función respectivamente (FOS).

- TABARG

Cuando se hace una llamada a un Macro se da una lista de parámetros actuales que serán reemplazados en el cuerpo

del Macro por los parámetros formales. Esta lista de parámetros actuales son los que se guardarán en la tabla de argumentos (TABARG).

Esta tabla tiene la siguiente forma:



Es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

- Los parámetros actuales de una Macro-llamada (REG1, REG2, REG3, REG4, REG5, REG6).

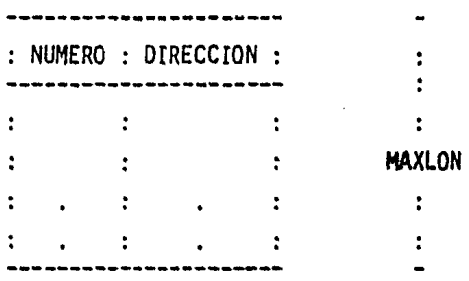
- TABSTACK

Esta tabla funciona como un stack dentro de MPAN, y es en este stack donde guardamos las direcciones para saber a dónde hay que retornar para seguir examinando el texto del archivo auxiliar. Recordando que el examen del texto es detenido al encontrar una Macro-llamada y una vez satisfecha esta llamada hay que continuar con la expansión que se estaba realizando. Todo este trabajo es sobre el archivo auxiliar.

Otra utilidad de esta misma tabla, es la de meter el valor numérico del primer registro de la tabla de - - -

Metavariables por medio del módulo PUSH y la de sacar del stack un valor numérico y asignarlo al segundo registro de la tabla de Metavariables por medio del módulo POP.

Esta tabla tiene la siguiente forma:



Es decir, es un arreglo de registros cuya dimensión es MAXLON. Los registros tienen los siguientes campos:

Este campo tiene un valor numérico asignado por el módulo PUSH (NUMERO).

- Este campo contiene las direcciones del archivo auxiliar (DIRECCIONES).

Descripción de variables

MPAN maneja internamente las siguientes variables:

- RESULT. Toma su valor de la función scanner:
  0 indica que es un símbolo cualquiera.
  1 indica que es un dígito.
  2 indica que es una palabra cualquiera.
  3 indica que es una palabra reservada.

4 indica que es el fin de línea.

5 indica que es un comentario.

9 indica que es el fin de archivo.

- IDENTI. Lo genera la función scanner, y es un registro que puede contener: una palabra cualquiera, una palabra reservada, un dígito o un símbolo cualquiera.

- LREG. Puede tomar dos valores por medio de la función scanner:

  El 0 indica que se tiene que leer un nuevo registro o línea de un archivo.

  El 1 indica que se va a continuar analizando la línea que ya tenemos en el buffer.

- BUFFER. Es un registro temporal que almacena una a una la línea de un archivo.

- BUFFAUX. Es un registro temporal que crea MPAN y lo enviará a alguno de los diferentes archivos de salida.

- AUXILIAR. Es un registro temporal que creará la función defmac o la función expmacros cuando uno de sus argumentos se encuentren entre paréntesis cuadrados.

- SWMAC. Puede tomar dos valores:

  1 nos indica si el nombre se encuentra en la tabla de los nombres de los Macros.

  0 nos indica que el nombre no se encuentra en la tabla de los nombres de los Macros.

- SWSUB. Puede tomar dos valores:
  1 nos indica si el nombre se encuentra en la tabla de los nombres de las subrrutinas o funciones.
  0 nos indica que el nombre no se encuentra en la tabla de nombres de las subrrutinas o funciones.

- SWNOM. Puede tener dos valores:
  1 nos indica si el nombre se encuentra en la tabla de las subrrutinas o funciones que se han llamado.
  0 nos indica si el nombre no se encuentra en la tabla de las subrrutinas o funciones que se han llamado.

Descripción de los mensajes de error

Existen 15 diferentes mensajes de error en MPAN, los cuales se describirán a continuación:

- Está mal estructurada la Macro-definición:

  La estructura de la Macro-definición es la siguiente:

```
%MACRO NOMBRE (ARG1,..., ARGN)     %MACRO NOMBRE ()
 CUERPO                             CUERPO
%ENDM                              %ENDM
```

Como podemos notar una Macro-definición puede tener argumentos o no pero siempre debe tener el nombre (NOMBRE) y después del NOMBRE siempre deben de existir los parëntesis redondos "()", si no caemos en este error.

- SOBRA UN ENDM

Como vimos anteriormente una Macro-definición está delimitada por las palabras reservadas %MACRO y %ENDM, si al estar analizando el texto fuente, nos encontramos un %ENDM sin antes existir la palabra %MACRO caeremos en este error.

- LA METAVARIABLE NO SE ENCUENTRA EN LA TABLA

La estructura de una Metavariable es la siguiente:

%METAVARIABLE = EXPRESION

Donde EXPRESION está compuesta por un valor numérico o por términos, pueden ser valores numéricos o Metavariables. Las Metavariables que componen una expresión deben estar en la tabla de Metavariables sino caemos en este error.

- REVISAR EL ARCHIVO AUXILIAR.ARC, LAS DIRECCIONES NO COINCIDEN CON LOS MACROS

En el archivo auxill.arc encontraremos primeramente los nombres de Macros con su dirección y su longitud (del Macro). Esas direcciones deben coincidir con el inicio del Macro del mismo nombre, sino habrá que corregirlo (la persona encargada) para que MPAN pueda encontrar el Macro-llamado.

- ESTA MAL ESTRUCTURADA LA MACRO-LLAMADA

La estructura de una Macro-llamada es la siguiente:

%NOMBRE (ARG1,..., ARGN) o %NOMBRE ()

Como podemos notar un Macro-llamado puede o no tener argumentos, pero después del nombre del Macro (NOMBRE) siempre debe llevar paréntesis redondo "()" sino caemos en este error.

- EN LA FUNCION PUSH, SE REBASO LA LONG. DEL STACK

  El índice del stack es mayor que MAXLON.

- EN LA FUNCION POP, EL APUNTADOR TOMO EL VALOR DE 0

  El índice del stack es igual a 0.

- LA SUBRRUTINA NO SE ENCUENTRA EN LA TABLA

  Cuando se hace un llamado a una subrrutina o función se revisan las subrrutinas o funciones que son llamadas dentro de su cuerpo, si esas subrrutinas o funciones no se encuentran en la tabla (significa que no existen en la Biblioteca de Análisis Numérico) caemos en este error.

- NO SE TECLEO NADA EN LA TERMINAL

  Al correr MPAN aparecerá el mensaje pidiéndo el nombre del archivo donde se encuentra el programa fuente y si el usuario no contesta nada y le da CR entonces aparecerá este mensaje.

Todos los mensajes que a continuación aparecerán, dependerán de la máquina computadora que se esté utilizando, si trabajas con la PDP-11/34 no tendrás problema alguno ya que esta máquina permite al usuario tener abiertos 6 archivos al mismo tiempo, de los cuales 3 emplea ella y el usuario puede emplear los otros 3. MPAN siempre mantiene abiertos 3 - -

archivos al mismo tiempo, sin tener ningún problema, pero si se cambiara de máquina que restringiera a menos de 3 para el usuario, se toparía con uno de estos mensajes:

- NO EXISTE LA BIBLIOTECA DE A. NUMERICO.
- NO EXISTE LA BIBLIOTECA DE MACROS.
- NO EXISTE EL ARCHIVO AUXILIAR.
- NO EXISTE EL ARCH. DEL USUARIO EN EL DIRECTORIO.
- NO EXISTE EL ARCHIVO AUXIL2.
- NO EXISTE EL ARCHIVO DE SALIDA. FTN.

5.3. Diagramas de flujo

En esta sección se encuentran los diagramas de flujo de las rutinas más importantes de MPAN.

a) Programa principal

b) Define Macros

c) Revisa tablas

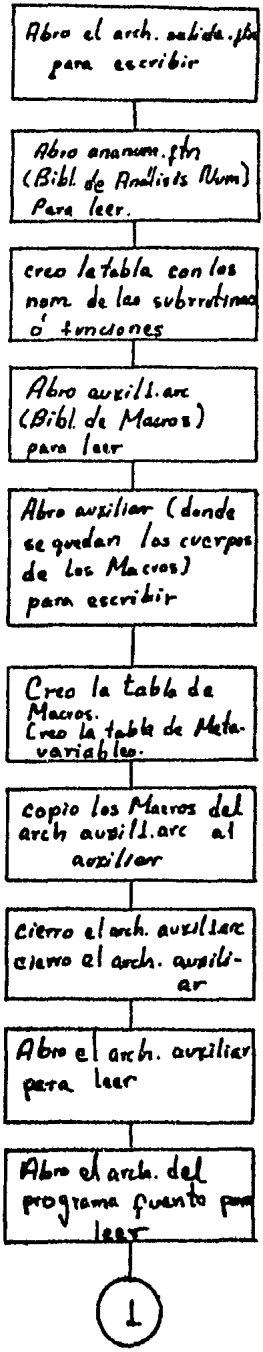
d) Expande subrrutinas

e) Reconocedor sintáctico

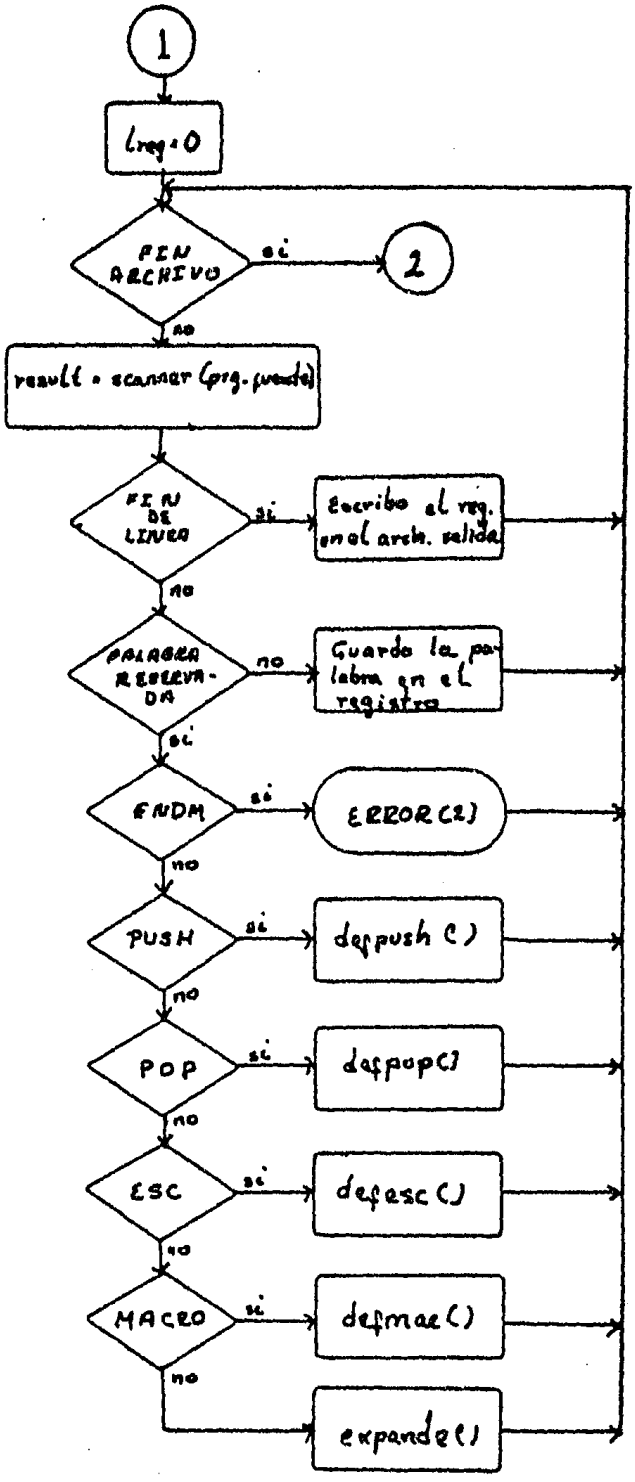
f) Reconoce

El Programa MPAN se encontrará al final de la tesis en el Anexo 1.

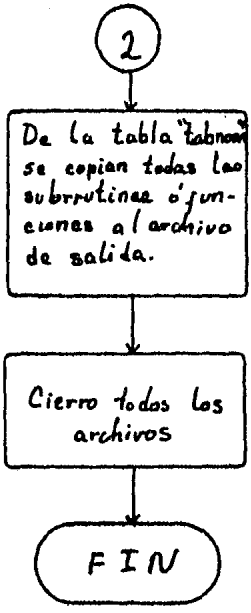
PROGRAMA PRINCIPAL



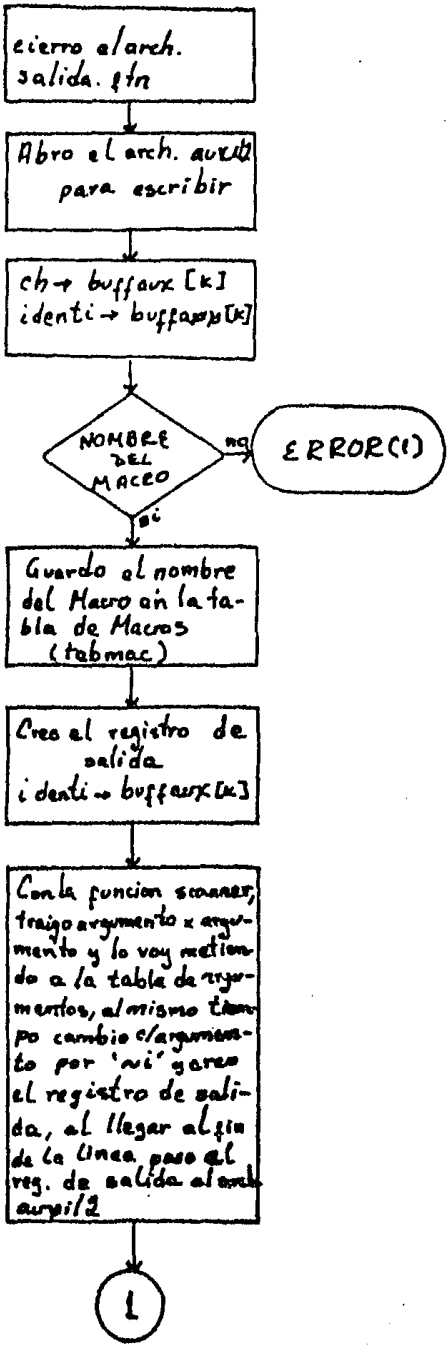
CONTINUA PROGRAMA PRINCIPAL



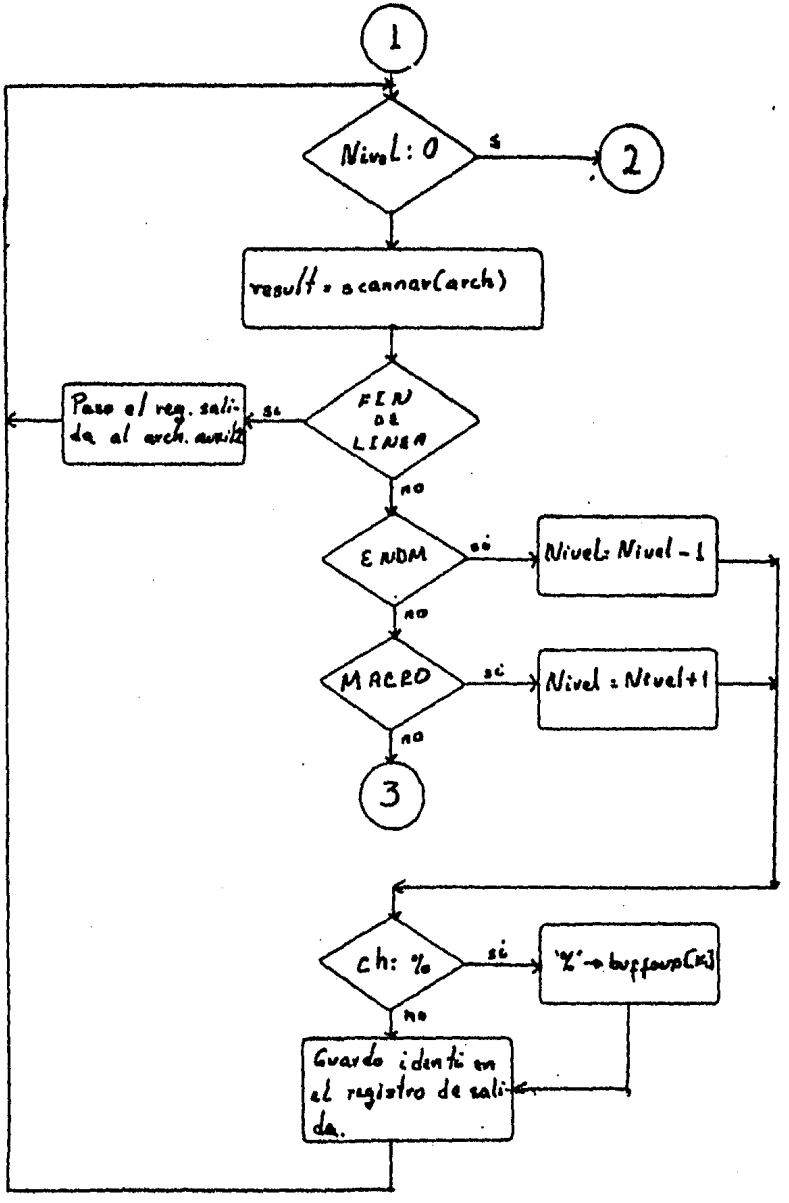
PROGRAMA PRINCIPAL

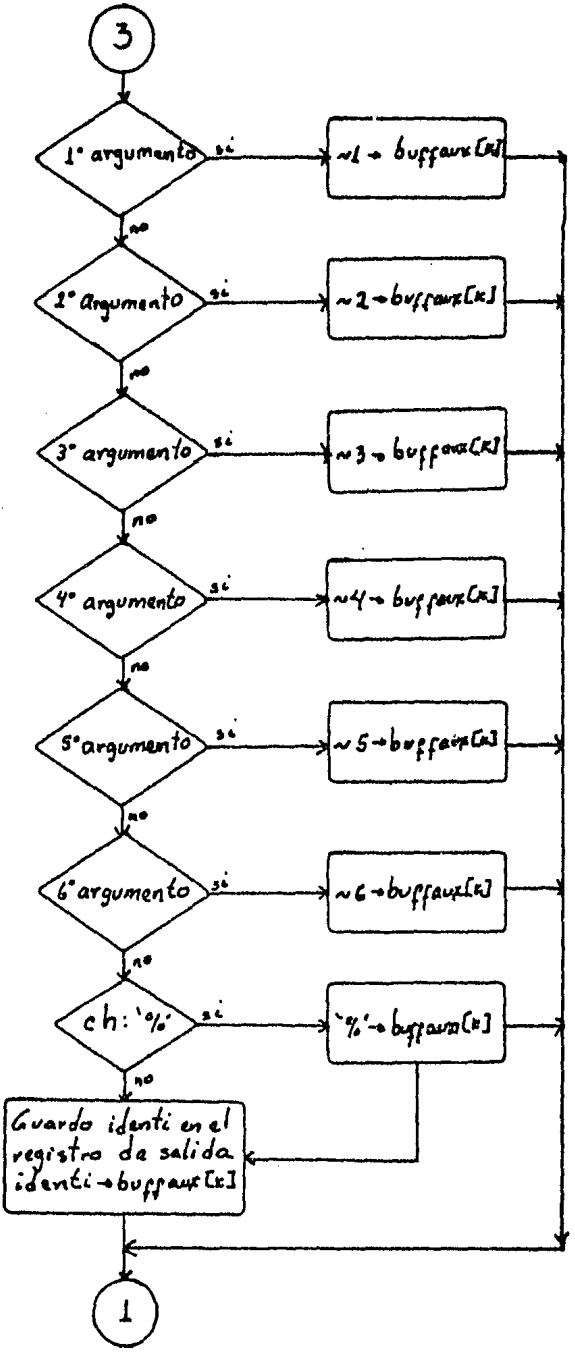


DEFINE MACROS

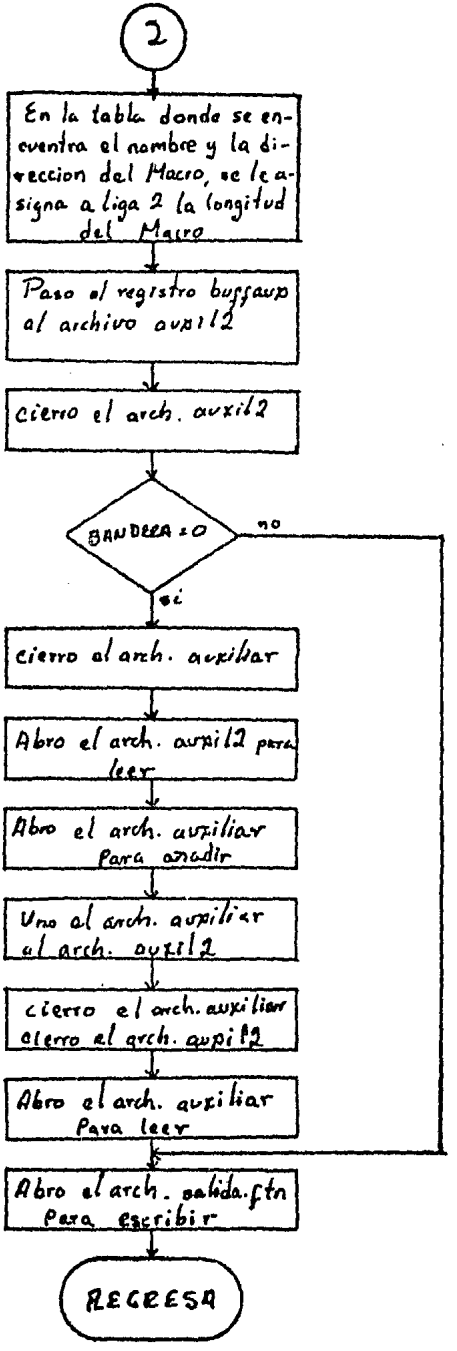


CONTINUA DEFINE MACROS

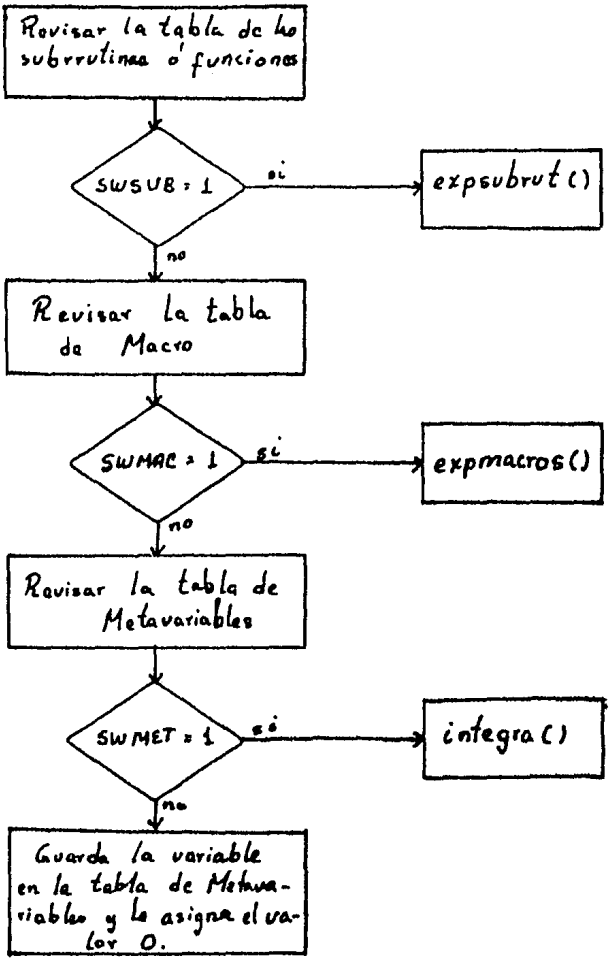



3
1° argumento
si
~1 → buffaux[k]
no
2° argumento
si
~2 → buffaux[k]
no
3° argumento
si
~3 → buffaux[k]
no
4° argumento
si
~4 → buffaux[k]
no
5° argumento
si
~5 → buffaux[k]
no
6° argumento
si
~6 → buffaux[k]
no
ch: '%'
si
'%' → buffaux[k]
no
Guardo identi en el registro de salida
identi → buffaux[k]
1

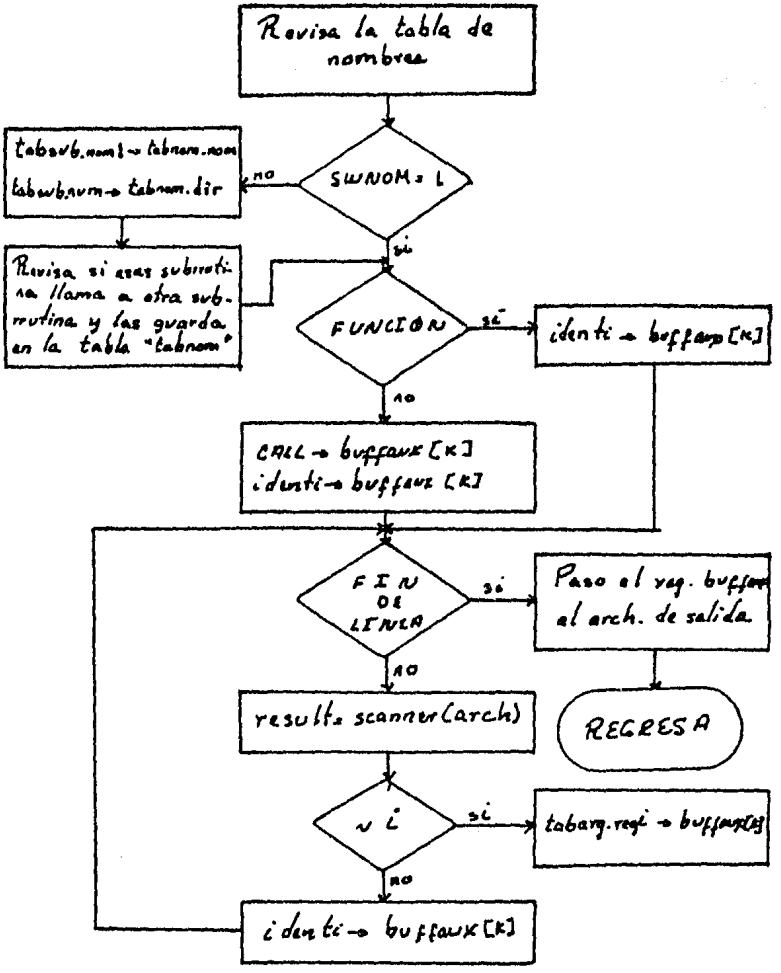
CONTINUA DEFINE MACROS



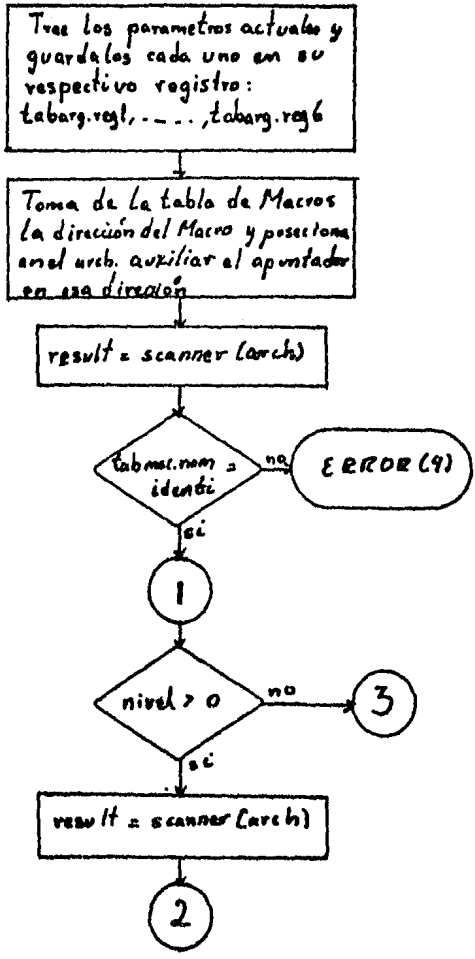
REVISA TABLAS

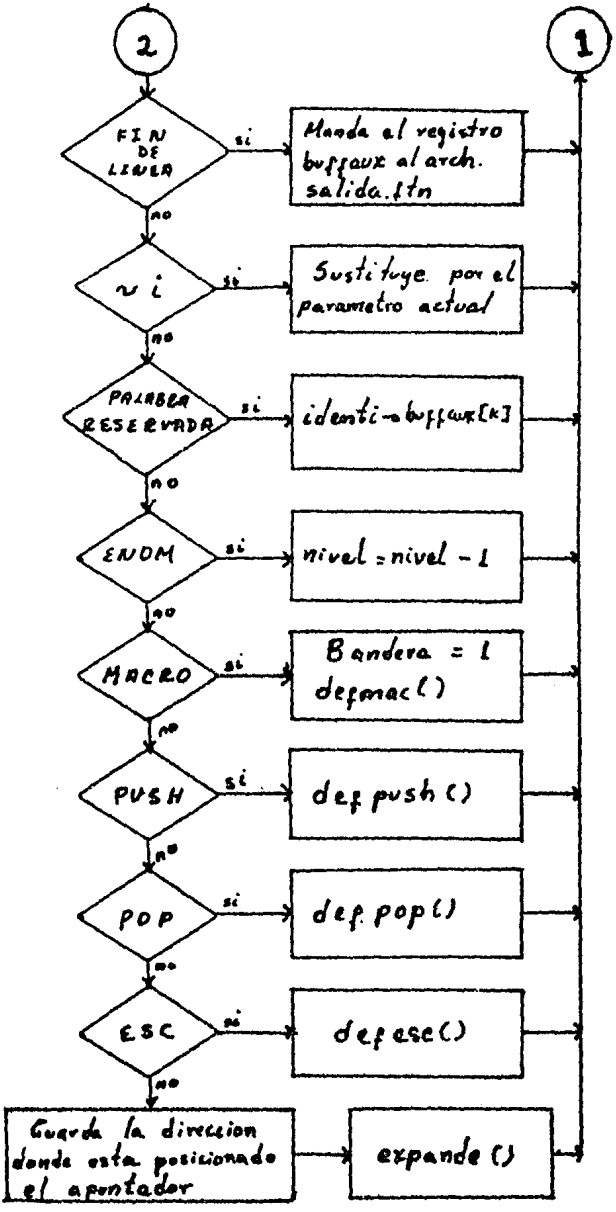


EXPANDE SUBRUTINAS

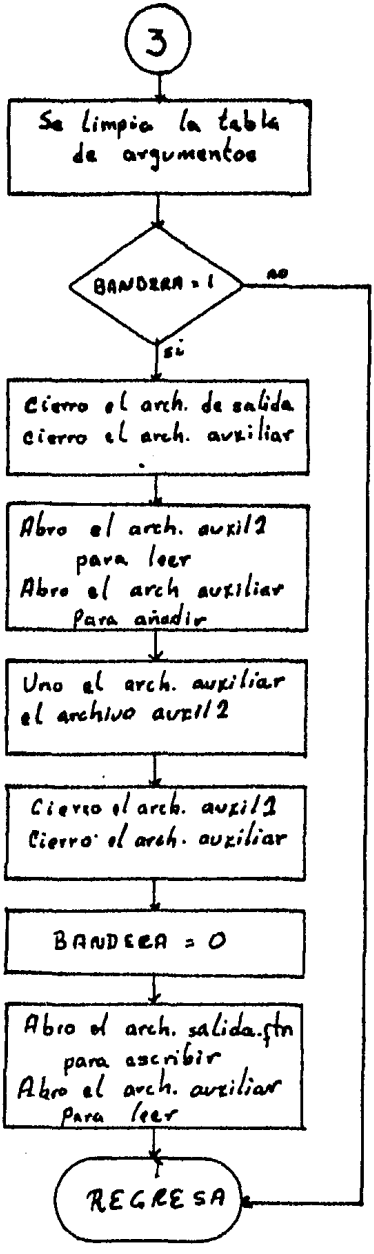


EXPANDE MACROS

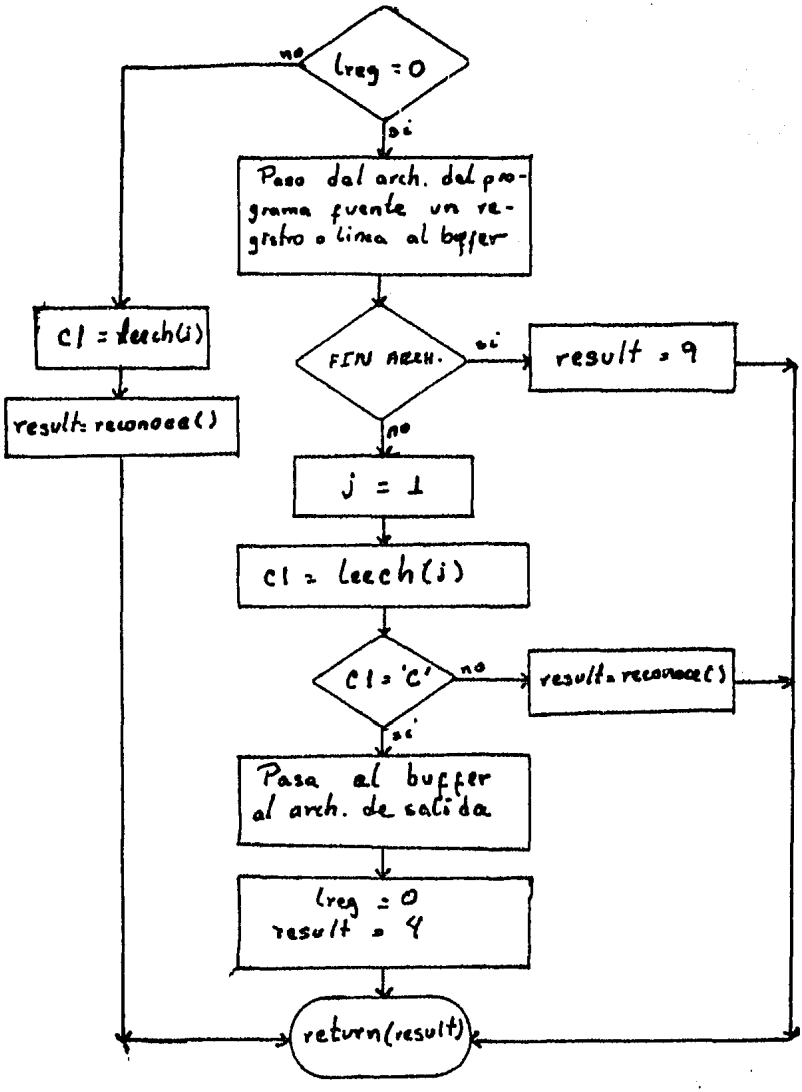



2
1
FIN DE LINEA
si
Manda al registro buffaux al arch. salida.ftn
no
~i
si
Sustituye por el parametro actual
no
PALABRA RESERVADA
si
identi=buffaux[k]
no
ENDM
si
nivel = nivel - 1
no
MACRO
si
Bandera = 1
defmac()
no
PUSH
si
def push()
no
POP
si
def pop()
no
ESC
si
defesc()
no
Guarda la direccion donde esta posicionado el apuntador
expande()

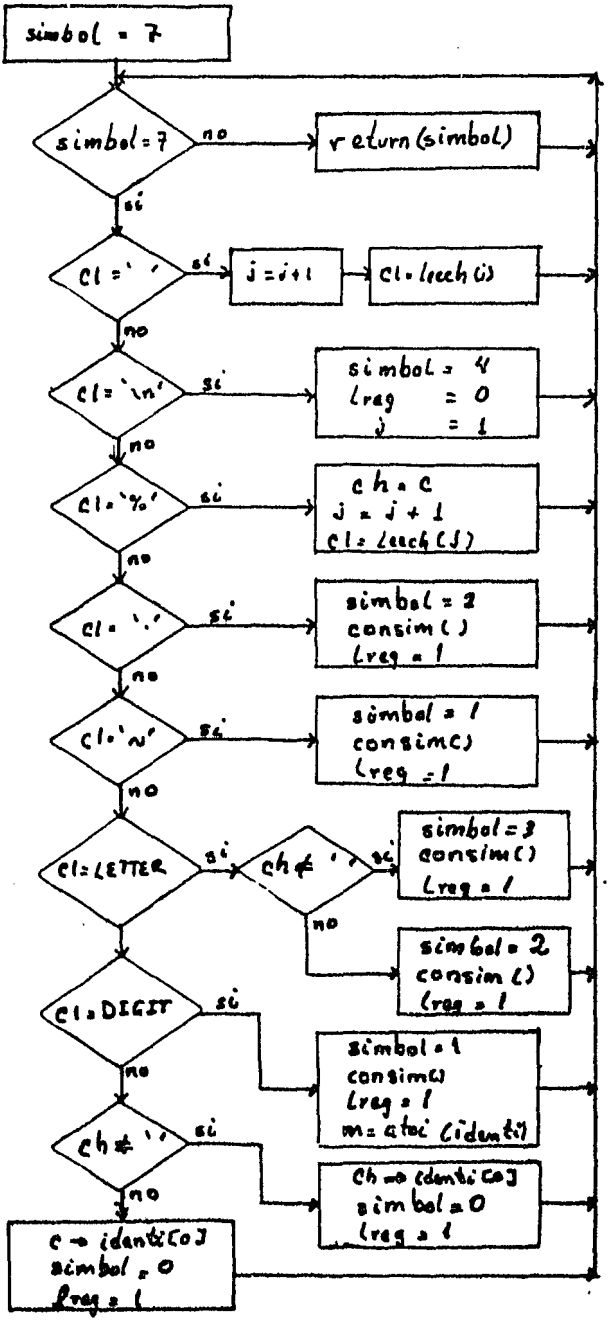
CONTINUA EXPANDE MACROS



RECONOCEDOR SINTACTICO



RECONOCEDOR



5.4. Conclusión

Para realizar este trabajo (MPAN), se eligió de entre varias máquinas del laboratorio de computación de la Facultad de Ciencias a la minicomputadora PDP-11/34, por ser multiusuario, contener varios lenguajes de programación, además de contar con un equipo periférico que da facilidad de uso en cualquier momento.

El lenguaje utilizado para construir MPAN fue "C". Se eligió "C" en lugar de Fortran y Pascal, por ser un lenguaje recursivo y más orientado a la programación de sistema.

Además de tener bien definidas la máquina y el lenguaje a utilizar se describió, de una manera más detallada, las tablas que maneja MPAN, los errores que detecta, las variables más importantes, detalles generales del programa.

VI. CONCLUSION

MPAN ha sido útil en el desarrollo de un medio ambiente adecuado para el diseño de algoritmos numéricos.

MPAN permite:

- Desarrollo sistemático de Bibliotecas de algoritmos.
- Expresar construcciones primitivas del Análisis Numérico en forma de patrones.
- Utilizar esos patrones de manera estructurada para crear otras construcciones no tan primitivas y que aumentan el grado de complejidad.
- Maquillar un lenguaje base, como Fortran, para incluir características como estructuras de control tipo Algol (while, repeat).

En resumen MPAN:

- Genera programas.
- Reduce el tamaño del programa.
- Mejora la sintaxis de un lenguaje.

VII. REFERENCIAS BIBLIOGRAFICAS

BARCUCI — BARCUCI ELENA Y PELACAVI GIANLUCA.
A SOFTWARE DEVELOPMENT SYSTEM BASED ON A MACROPROCESSOR
SOFTWARE PRACTICE AND EXPERIENCE
JOHN WILEY AND SONS, LTD
VOL. 14, No. 6, PAG. 519-531, JUNIO 1984

BROWN — BROWN P.J.
MACRO PROCESSORS
JOHN WILEY AND SONS, 1975.

BROWN-2 — BROWN DAVID C.
ADD MACRO-EXPANSION TO YOUR MICROCOMPUTER
BYTE PUBLICATION INC
OCTOBER 1980.

DONGARRA — DONGARRA J.J., MOLER C.B., BUNCH J.R. STEWART G.W.
LINPACK USER'S GUIDE
SIAM PHILADELPHIA 1979.

GRIES — GRIES DAVID.
COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS
JOHN WILEY AND SONS, INC, 1971.

KERNIGHAN — KERNIGHAN AND PLAUGER.
SOFTWARE TOOLS
READING, MASS, ADISSON-WESLEY, 1976.

KERNIGHAN-2 — KERNIGHAN BRIAN W. Y RITCHIE DENNIS M.
THE "C" PROGRAMING LANGUAGE
BELL LABORATORIES, INCORPORATED 1978.

LEVINE — LEVINE GUILLERMO.
FOREST: PROCESADOR PARA EXTENDER FORTRAN:
MEMORIAS DE LA CONFERENCIA SOBRE MICROCOMPUTADORAS Y MICROPROCESADORES
FUNDACION ARTURO R.
ABRIL 1980.

MC CALL — MC CALL THOMAS RICHARD.
INTRODUCTION TO NUMERICAL METHODS AND FORTRAN PROGRAMMING
JOHN WILEY AND SONS, INC, 1967.

SASSA — SASSA MASATAKA.
A PATTERN MATCHING MACRO PROCESSOR.
SOFTWARE-PRACTICE AND EXPERIENCE.
JOHN WILEY AND SONS, LTD
VOL 9, PAG. 439-456, 1979.

SOMMERVILLE — SOMMERVILLE IAN.
A PATTERN MATCHING SYSTEM.
SOFTWARE PRACTICE AND EXPERIENCE
JOHN WILEY AND SONS, LTD
VOL. 12, PAG. 517-530, 1982.

VIII. A P E N D I C E

A N E X O 1

```
/*  */
/* EL MACRO-PROCESADOR DE ANALISIS NUMERICO ES UNA HERRAMIENTA UTIL
 */
/* PARA LA RESOLUCION DE PROBLEMAS DE ANALISIS NUMERICO
 */
/*  */
 include <stdio.h>
 define MAXLON    30
 define MAXLIN    80
 define LONGITUD  100
 define LETTER   'a'
 define DIGIT    '0'
int lres, j, k, w, n, p, s, ss, x, nn, pp, xx, sp, ch, bandera, st;
int m, aoe, ppp, kk, sd, y, hh;
char buffer[LONGITUD], identi[MAXLON], buffaux[LONGITUD];
char auxiliar[MAXLON];
FILE *fp1, *fp2, *fp3, *fan, *fes, *fres, *faux1, *faux2;
struct regcer {
            char nombre[MAXLON];
            char arg1[MAXLON];
            char arg2[MAXLON];
            char arg3[MAXLON];
            char arg4[MAXLON];
            char arg5[MAXLON];
            char arg6[MAXLON];
          r long liga1;
            long liga2;
               } tabmac[MAXLIN];
struct reguno {
            char nom1[MAXLON];
            char nom2[MAXLON];
            char nom3[MAXLON];
            char nom4[MAXLON];
            long num;
            int fos;
               } tabsub[MAXLON];
struct regdos {
            char nombre[MAXLON];
            int numero;
               } tabmet[MAXLON];
struct regtre {
            char nombre[MAXLON];
            long dir;
            int fos;
               } tabnom[MAXLON];
struct regcua {
           int numero;
          long direccion;
               } tabstack[MAXLON];
struct regcin {
```

```
            char res1[MAXLON];
            char res2[MAXLON];
            char res3[MAXLON];
            char res4[MAXLON];
            char res5[MAXLON];
            char res6[MAXLON];
                } tabars[MAXLON];
/*  */
/* PROGRAMA PRINCIPAL: CREA TABLAS (MACROS, METAVARIABLES, SUBRRU */
/* TINAS o FUNCIONES), ANALIZA EL TEXTO FUENTE (ESCRITO POR EL -- */
/* USUARIO) Y DEPENDIENDO DE ESE ANALIZIS LLAMA A DIFERENTES PRO- */
/* PROGRAMAS.                                                     */
/*  */
main()
  {
    FILE *fopen(), *fclose();
    char *gets(), *fputs();
    char nomar[MAXLON];
    int result, c, ind, clav1;
    clear(buffaux, LONGITUD);
    clear(identi, MAXLON);
    clear(buffer, LONGITUD);
    bandera = 0, hh = 0; st = 1; sd = 0;
    x = 1; clav1 = 0; y = 0; aoe = 0;
    printf("Teclea el nombre del programa fuente\n");
    if((gets(nomar)) == NULL)
        errrr(9);
    fp2 = fopen("salida.ftn", "w");
    printf("El programa en FORTRAN se encuentra en el arch.
 salida.ftn\n");
    printf("si deseas que ese archivo contenga las subrrutinas\n");
    printf("teclea 'S' en caso contrario 'N'\n");
    c = toupper(getchar());
    if( c == 'S' )
      aoe = 1;
    if((fan = fopen("ananum.ftn", "r")) == NULL)
        errrr(10);
    tablas(fan);
    fclose(fan);
    if((faux1 = fopen("auxil1.arc", "r")) == NULL)
        errrr(11);
    fp3 = fopen("auxiliar", "w");
    dostab();
    copyarc(faux1, fp3);
    fclose(fp3);
    fclose(faux1);
    if((fp3 = fopen("auxiliar", "r")) == NULL)
        errrr(12);
    if((fp1 = fopen(nomar, "r")) == NULL)
        errrr(13);
    lreg = 0;
    do {
        result = scanner(fp1);
```

```
    switch(result) {
    case 3 : if((strcmp(identi, "MACRO")) == 0)
                {
                  defmac(fp1); lreg = 0;
                }
             else
             if((strcmp(identi, "ENDM")) == 0)
               errrr(2);
             else
             if((strcmp(identi, "PUSH")) == 0)
               {
               defpush(); clav1 = 1;
               }
             else
             if((strcmp(identi, "POP")) == 0)
               {
               defpop(); clav1 = 1;
               }
             else
             if((strcmp(identi, "ESC")) == 0)
               defesc();
             else
                expande(fp1);
             break;
     case 4 : if( clav1 != 1 )
               {
             buffaux[k] = 'n'; k++;
             buffaux[k] = '0';
             fputs(buffaux, fp2);
             clear(buffaux, LONGITUD);
               }
             else
               {
                 clav1 = 0; clear(buffaux, LONGITUD);
               }
             break;
     case 5 : break;
     default : if( result != 9 ) {
                ind = creaaux(identi, buffaux, k); k = ind;
                                  }
               break;
               }
  }
while( result != 9 );
fclose(fp1);
fclose(fp3);
if( aoe == 1 )
  {
    if((fan = fopen("ananum.ftn", "r")) == NULL)
        errrr(10);
    subrrut(fan);
    fclose(fan);
  }
```

```
   fclose(fp2);
  }
/*  */
/*      R E C O N O C E D O R    S I N T A C T I C O
 */
/*  */
scanner(arch)
FILE *arch;
  {
    int c, n1;
    char *fgets(), *fputs();
    ch = ' ';
    if( lreg != 1 )
      {
        clear(buffer, LONGITUD);
        if((fgets(buffer, LONGITUD, arch)) == NULL)
            n1 = 9;
        else
          {
            j = 1;
            c = leech();
            if( c == 'c' || c == 'C' )
              {
                fputs(buffer, fp2);
                lreg = 0; n1 = 5;
                clear(buffer, LONGITUD);
                clear(identi, MAXLON);
              }
            else
              n1 = reconoce(c);
          }
      }
    else
      {
        c = leech();
        n1 = reconoce(c);
      }
    return(n1);
  }
reconoce(c)
int c;
  {
    int simbol;
    int atoi();
    clear(identi, MAXLON);
    ch = ' '; simbol = 7;
    if( lreg == 0 )
      k = 0;
    do {
        switch(type(c)) {
        case ' ' : if( lreg == 0 )
                     {
                       buffaux[k] = ' '; k++;
```

```
                    }
                    J++; c = leech();
                    break;
            case 'n' : simbol = 4; lreg = 0; J = 1;
                     break;
            case '%' : ch = c; J++; c = leech();
                    break;
            case ',' : simbol = 2; consim(); lreg = 1;
                    break;
            case '=' : simbol = 1; consim(); lreg = 1;
                    break;
            case LETTER : if( ch != ' ' )
                            simbol = 3;
                          else
                            simbol = 2;
                          lreg = 1; consim();
                          break;
           case DIGIT : simbol = 1; lreg = 1; consim();
                          m = atoi(identi);
                          break;
           default : if( ch != ' ' )
                       identi[0] = ch;
                     else
                       identi[0] = c;
                     simbol = 0; lreg = 1; identi[1] = '0';
                     J++; break;
                    }
        }
      while(simbol == 7);
      return(simbol);
    }
  leech()
    {
      return(buffer[J - 1]);
    }
  type(c)
  int c;
    {
      if(c >= 'a'  c <= 'z' || c >= 'A'  c <= 'Z')
        return(LETTER);
      else
      if(c >= '0'  c <= '9')
        return(DIGIT);
      else
        return(c);
    }
 toupper(c)
 int c;
   {
     return((c >= 'a'  c <= 'z') ? c - ('a' - 'A') : c);
   }
consim()
  {
```

```
    int i, c;
    c = leech();
    i = 0;
    if( type(c) == '"' )
      {
        identi[i] = c;
        ++J; ++i;
        c = leech();
      }
    while(type(c) == LETTER || type(c) == DIGIT || type(c) == '.')
      {
        identi[i] = c;
        ++J; ++i;
        c = leech();
      }
    identi[i] = '0';
  }
clear(a, l)
char a[];
int l;
  {
    int i;
    for(i = 0; i < l; ++i)
      a[i] = ' ';
  }
tablas(ap)
FILE *ap;
  {
    int sw, result, sw1;
    s = 1; sw = 0; sw1 = 0; lreg = 0;
    do {
        result = scanner(ap);
        switch(result) {
        case 1 : ++sw1;
                 switch(sw1) {
                 case 1 : tabsub[s].num = m; break;
                 case 2 : tabsub[s].fos = m; break;
                 default : break;
                           }
                 break;
        nti); break;
                 case 2 : llenatab(tabsub[s].nomak;
                 case 3 : llenatab(tabsub[s].nom3, identi); break;
                 case 4 : llenatab(tabsub[s].nom4, identi); break;
                 default : break;
                           }
                   break;
          case 4 : s++; sw1 = 0; sw = 0;
                   break;
          default : break;
                    }
```

```
      }
    while( result != 0 );
  }
dostab()
  {
    int result, sw;
    n = 1; p = 1; lreg = 0; sw = 0;
    do {
        result = scanner(faux1);
        switch(result) {
        case 1 : sw++;
                 switch(sw) {
                 case 1 : tabmac[n].lisa1 = m; break;
                 case 2 : tabmac[n].lisa2 = m; break;
                 default : break;
                           }
                 break;
        case 2 : llenatab(tabmac[n].nombre, identi);
                 break;
        case 4 : n++; sw = 0;
                 break;
        default : break;
                  }
       }
     while( result != 0 );
     lreg = 0;
     do {
         result = scanner(faux1);
         switch(result) {
         case 1 : tabmet[p].numero = m; break;
         case 2 : llenatab(tabmet[p].nombre, identi);
                  break;
         case 4 : p++; break;
                  }
       }
      while( result != 0 );
  }
copyarc(ent, sal)
FILE *ent, *sal;
  {
    char *fputs();
    int result;
    lreg = 0;
    result = scanner(ent);
   do {
        while( result == 5  lreg == 0 )
          result = scanner(ent);
        if((strcmp(identi, "END")) == 0)
          result = 9;
        else
          {
            fputs(buffer, sal);
            lreg = 0;
```

```
        }
      if( result != 9 )
        result = scanner(ent);
    }
  while( result != 9 );
}
/*  */
/* ESTE PROGRAMA ES LLAMADO CUANDO EXISTE UNA MACRO-DEFINISION
/*  */
defmac(arch)
FILE *arch;
{
  FILE *fopen(), *fclose();
  char *fputs();
  int nivel, result, contad, ind;
  nivel = 1; contad = 1;
  fclose(fp2);
  faux2 = fopen("auxil2", "w");
  buffaux[k] = ch; k++;
  ind = creaaux(identi, buffaux, k); k = ind;
  result = scanner(arch);
  if( result == 2 )
    {
      llenatab(tabmac[n].nombre, identi);
      ind = creaaux(identi, buffaux, k); k = ind;
      setpar(arch);
      contad++;
      while( nivel > 0 )
        {
          result = scanner(arch);
          if( result == 4  lres == 0 )
            {
              buffaux[k] = 'n'; k++;
              buffaux[k] = '0';
              fputs(buffaux, faux2);
              contad++;
              clear(buffaux, LONGITUD);
            }
          else
            {
              if((strcmp(identi, "ENDM")) == 0)
                {
                  nivel--; k++;
                  buffaux[k] = '%'; k++;
                  ind = creaaux(identi, buffaux, k); k = ind;
                }
              else
              if((strcmp(identi, "MACRO")) == 0)
                {
                  nivel++; k++;
                  buffaux[k] = '%'; k++;
                  ind = creaaux(identi, buffaux, k); k = ind;
                }
```

```
        else
        if((strcmp(identi, tabmac[n].arg1)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '1'; k = k + 2;
          }
        else
        if((strcmp(identi, tabmac[n].arg2)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '2'; k = k + 2;
          }
        else
        if((strcmp(identi, tabmac[n].arg3)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '3'; k = k + 2;
          }
        else
        if((strcmp(identi, tabmac[n].arg4)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '4'; k = k + 2;
          }
        else
        if((strcmp(identi, tabmac[n].arg5)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '5'; k = k + 2;
          }
        else
        if((strcmp(identi, tabmac[n].arg6)) == 0)
          {
            buffaux[k] = '~'; k++;
            buffaux[k] = '6'; k = k + 2;
          }
        else
        if(ch == '%')
          {
            buffaux[k] = '%'; k++;
            ind = creaaux(identi, buffaux, k); k = ind;
          }
        else
          {
            ind = creaaux(identi, buffaux, k); k = ind;
          }
      }
  }
tabmac[n].liga2 = contad;
n++;
buffaux[k] = 'n'; k++;
buffaux[k] = '0';
fputs(buffaux, faux2);
```

```
        clear(buffaux, LONGITUD);
        fclose(faux2);
        if( bandera == 0 )
          {
            fclose(fp3);
            if((faux2 = fopen("auxil2", "r")) == NULL)
                errrr(14);
            if((fp3 = fopen("auxiliar", "a")) == NULL)
                errrr(12);
            copyarc(faux2, fp3);
            fclose(fp3);
            fclose(faux2);
            if((fp3 = fopen("auxiliar", "r")) == NULL)
                errrr(12);
          }
        if((fp2 = fopen("salida.ftn", "a")) == NULL)
            errrr(15);
      }
    else
    errrr(1);
  }
creaaux(s, t, ind)
char s[], t[];
int ind;
  {
    int i, ind1;
    i = 0;
    ind1 = ind;
    while( s[i] != '0' )
      {
        t[ind] = s[i];
        ind++; ++i;
      }
    if( ind1 == 0  hh == 0 )
      ind = 6;
    hh = 0;
    t[ind] = ' ';
    ++ind;
    return(ind);
  }
getpar(arch)
FILE *arch;
  {
    int sw, result, nivel1, sennal, ind;
    long m1;
    char *fputs();
    sennal = 0; kk = 0;
    sw = 0; result = scanner(arch);
    if( identi[0] == '('  result == 0 )
      {
        buffaux[k] = '('; k++;
        result = scanner(arch);
        while(identi[0] != ')'  result != 4
```

```
{
  if(identi[0] == '['  lreg == 0)
    {
      sennal = 1; nivel1 = 1;
      while( nivel1 > 0 )
        {
          result = scanner(arch);
          if(identi[0] == '['  lreg == 0)
            nivel1++;
          else
          if(identi[0] == ']'  lreg == 0)
            {
            nivel1--;
            auxiliar[kk] = '0';
            }
          else
            {
            ind = creaaux(auxiliar, identi, kk); kk = ind;
          }
        }
    }
  if( identi[0] != ',' )
    {
      sw++;
      switch(sw) {
      case 1 : movarg(tabmac[n].arg1, sw, sennal); break;
      case 2 : movarg(tabmac[n].arg2, sw, sennal); break;
      case 3 : movarg(tabmac[n].arg3, sw, sennal); break;
      case 4 : movarg(tabmac[n].arg4, sw, sennal); break;
      case 5 : movarg(tabmac[n].arg5, sw, sennal); break;
      case 6 : movarg(tabmac[n].arg6, sw, sennal); break;
      default : break;
                }
      sennal = 0;
    }
   if( identi[0] == ','  result == 0 )
       {
         buffaux[k] = ','; k = k + 2;
       }
    result = scanner(arch);
  }
n = n - 1;
m1 = tabmac[n].liga1 + tabmac[n].liga2;
n++;
tabmac[n].liga1 = m1;
if( identi[0] == ')'  result == 0 )
  {
    buffaux[k] = ')'; k++;
    result = scanner(arch);
  }
else
  {
    buffaux[k] = 'n'; k++;
```

```
            buffaux[k] = '0';
            fputs(buffaux, faux2);
          }
      }
     if( result == 4  lres == 0 )
       {
         buffaux[k] = 'n'; k++;
         buffaux[k] = '0';
         fputs(buffaux, faux2);
         clear(buffaux, LONGITUD);
       }
     else
       errrr(1);
   }
movars(s, l, ll)
char s[];
int l, ll;
  {
    if( ll == 1 )
      llenatab(s, auxiliar);
    else
      {
        llenatab(s, identi);
        buffaux[k] = '~'; k++;
        switch(l) {
        case 1 : buffaux[k] = '1'; k++; break;
        case 2 : buffaux[k] = '2'; k++; break;
        case 3 : buffaux[k] = '3'; k++; break;
        case 4 : buffaux[k] = '4'; k++; break;
        case 5 : buffaux[k] = '5'; k++; break;
        case 6 : buffaux[k] = '6'; k++; break;
        default : break;
                  }
      }
  }
llenatab(s, t)
char s[], t[];
  {
    int i;
    for( i = 0; i < MAXLON; i++ )
       s[i] = t[i];
  }
/*  */
/* ESTE PROGRAMA ES LLAMADO CUANDO EXISTE UNA MACRO-LLAMADA. Y SE EN
 */
/* CARGA DE REVISAR LAS TABLAS (MACROS, SUBRRUTINAS O FUNCIONES, ME-
 */
/* TAVARIABLES), PARA LLAMAR A LOS DIFERENTES PROGRAMAS QUE TRATAN -
 */
/* CADA UNO DE LOS CASOS
 */
/*  */
expande(arch)
```

```
  FILE *arch;
    {
      if(( revsub(identi)) == 1)
          expsub();
      else
      if((revmac()) == 1)
        {
          y++;
          expmacros(arch);
          lres = 0;
        }
      else
          evalua(arch);
    }
  revsub(s)
  char s[];
    {
      int sw, resmac;
      ss = 1;
      do {
          if((resmac = strcmp(s, tabsub[ss].nom1)) == 0)
            ;
          else
            ss++;
        }
      while( ss < s  resmac != 0 );
      if( resmac == 0 )
        sw = 1;
      else
        sw = 0;
      return(sw);
    }
  revmac()
    {
      int sw, resmac;
      nn = 1;
      do {
          if((resmac = strcmp(identi, tabmac[nn].nombre)) == 0)
            ;
          else
            nn++;
        }
      while( nn < n  resmac != 0 );
      if( resmac == 0 )
        sw = 1;
      else
        sw = 0;
      return(sw);
    }
revmet()
  {
    int sw, resmac;
    pp = 1;
```

```
    do {
        if((resmac = strcmp(identi, tabmet[pp].nombre)) == 0)
           ;
        else
          pp++;
       }
    while( pp < p  resmac != 0 );
    if( resmac == 0 )
      sw = 1;
    else
      sw = 0;
    return(sw);
  }
revnom()
  {
    int sw, resmac;
    xx = 1;
    do {
        if((resmac = strcmp(identi, tabnom[xx].nombre)) == 0)
          ;
        else
          xx++;
       }
    while( xx < x  resmac != 0 );
    if( resmac == 0 )
      sw = 1;
    else
      sw = 0;
    return(sw);
  }
/*  */
/* ESTE PROGRAMA REVISA EL CASO, CUANDO LA MACRO-LLAMADA SE TRATA DE
 */
/* UNA SUBRRUTINA O FUNCION
 */
/*  */
expsub()
  {
    int result, resiv, ind, ii;
    char *fputs();
    ii = tabsub[ss].fos;
    if((revnom()) == 1)
      {
      if( ii == 1 ) {
        buffaux[7] = 'C'; buffaux[8] = 'A'; buffaux[9] = 'L';
        buffaux[10] = 'L'; k = 12;
                      }
       ind = creaaux(identi, buffaux, k); k = ind;
     }
   else
     {
       llenatab(tabnom[x].nombre, tabsub[ss].nom1);
       tabnom[x].dir = tabsub[ss].num;
```

```
      tabnom[x].fos = tabsub[ss].fos;
      x++;
      resiv = ss;
      if(( novacio(tabsub[resiv].nom2)) != 0 )
        subnom2(tabsub[resiv].nom2);
      if(( novacio(tabsub[resiv].nom3)) != 0 )
        subnom2(tabsub[resiv].nom3);
      if(( novacio(tabsub[resiv].nom4)) != 0 )
        subnom2(tabsub[resiv].nom4);
     if( ii == 1 )  {
      buffaux[7] = 'C'; buffaux[8] = 'A'; buffaux[9] = 'L';
      buffaux[10] = 'L'; k = 12;
                    }
      ind = creaaux(tabsub[resiv].nom1, buffaux, k); k = ind;
    }
  result = scanner(fp1);
  while( result != 4  lreg != 0 )
    {
      if( identi[0] == '~' )
        {
          switch(identi[1]) {
          case '1' : ind = creaaux(tabars[y].reg1, buffaux, k);
                     k = ind; break;
          case '2' : ind = creaaux(tabars[y].reg2, buffaux, k);
                     k = ind; break;
          case '3' : ind = creaaux(tabars[y].reg3, buffaux, k);
                     k = ind; break;
          case '4' : ind = creaaux(tabars[y].reg4, buffaux, k);
                 k = ind; break;
          case '6' : ind = creaaux(tabars[y].reg6, buffaux, k);
                     k = ind; break;
          default : break;
                              }
        }
      else
        {
        ind = creaaux(identi, buffaux, k); k = ind;
        }
      result = scanner(fp1);
    }
  buffaux[k] = 'n'; k++;
  buffaux[k] = '0';
  fputs(buffaux, fp2);
  clear(buffaux, LONGITUD);
  }
subnom2(s)
char s[];
  {
    llenatab(tabnom[x].nombre, s);
    if((revsub(s)) == 1)
      {
```

```
        tabnom[x].dir = tabsub[ss].num;
        tabnom[x].fos = tabsub[ss].fos;
        x++;
      }
    else
      errrr(8);
  }
novacio(s)
char s[];
  {
    int i, sw;
    sw = 0;
    for( i = 0; i < MAXLON; i++ )
       {
         if( s[i] != ' '  s[i] != '0' )
           sw = 1;
       }
    return(sw);
  }
/*  */
/* ESTE PROGRAMA EXPANDE EL MACRO, CUANDO SE REALIZO UNA MACRO-LLA-
 */
/* MADA.
 */
/*  */
expmacros(arch)
FILE *arch;
  {
    char *fputs();
    FILE *fopen(), *fclose();
    int nivel, result, clav1, ind, nivel1;
    long resdir;
    nivel = 1;
    getres(arch);
    lreg = 0; clav1 = 0;
    resdir = tabmac[nn].lisa1;
    lseek(fp3, tabmac[nn].lisa1, 0);
    result = scanner(fp3);
    if( result == 3 )
      {
        result = scanner(fp3);
        if((strcmp(identi, tabmac[nn].nombre)) == 0)
          {
            lreg = 0; resdir++;
            while( nivel > 0 )
              {
                result = scanner(fp3);
                if( result == 4  lreg == 0 )
                  {
                    if( clav1 != 1 )
                      {
                        buffaux[k] = 'n'; k++;
                        buffaux[k] = '0';
```

```
                    fputs(buffaux, fp2);
                    clear(buffaux, LONGITUD);
                  }
                else
                  {
                    clav1 = 0;
                    clear(buffaux, LONGITUD);
                  }
                resdir++;
              }
            else
            if( result == 3 )
              {
                if((strcmp(identi, "ENDM")) == 0)
                  nivel--;
                else
                if((strcmp(identi, "MACRO")) == 0)
                  {
                    bandera = 1;
                    defmac(fp3);
                    lreg = 0;
                  }
                else
                if((strcmp(identi, "PUSH")) == 0)
                  {
                  defpush(); clav1 = 1;
                }
                else
                if((strcmp(identi, "POP")) == 0)
                  {
                  defpop(); clav1 = 1;
                  }
                else
                if((strcmp(identi, "ESC")) == 0)
                  defesc();
                else
                  {
                  sd++; tabstack[sd].direccion = resdir; nivel1 =
nivel;
                  expande(fp3); resdir = tabstack[sd].direccion;
resdir++;
                  sd--; lseek(fp3, resdir, 0); nivel = nivel1;
                }
              }
            else
            if( identi[0] == '~' )
              {
                switch(identi[1]) {
                case '1' : ind = creaaux(tabarg[y].reg1, buffaux,
k);
                           k = ind; break;
                case '2' : ind = creaaux(tabarg[y].reg2, buffaux,
k);
```

```
                                k = ind; break;
    k);                 case '3' : ind = creaaux(tabars[y].reg3, buffaux,
                                k = ind; break;
    k);                 case '4' : ind = creaaux(tabars[y].reg4, buffaux,
                                k = ind; break;
    k);                 case '5' : ind = creaaux(tabars[y].reg5, buffaux,
                                k = ind; break;
    k);                 case '6' : ind = creaaux(tabars[y].reg6, buffaux,
                                k = ind; break;
                        default : break;
                                        }
                      }
                    else
                      {
                         ind = creaaux(identi, buffaux, k); k = ind;
                       }
                }
              limpia();
              y--;
              if( bandera == 1 )
                {
                  fclose(fp2); fclose(fp3);
                  if((faux2 = fopen("auxil2", "r")) == NULL)
                      errrr(14);
                  if((fp3 = fopen("auxiliar", "a")) == NULL)
                      errrr(12);
                  copyarc(faux2, fp3);
                  fclose(fp3);
                  fclose(faux2);
                  bandera = 0;
                  if((fp2 = fopen("salida.ftn", "a")) == NULL)
                      errrr(15);
                  if((fp3 = fopen("auxiliar", "r")) == NULL)
                      errrr(12);
                }
            }
          else
            errrr(4);
        }
      else
        errrr(4);
    }
getres(arch)
FILE *arch;
  {
    int result, sw, sennal, nivel1, ind;
    sennal = 0; sw = 0; kk = 0;
    result = scanner(arch);
    if( identi[0] == '(' )
```

```
    {
    result = scanner(arch);
  while( identi[0] != ')'  result != 4  lres != 0 )
    {
      if( identi[0] == '['  result == 0 )
        {
          sennal = 1; nivel1 = 1;
          while( nivel1 > 0 )
            {
              result = scanner(arch);
              if( identi[0] == '['  result == 0 )
                nivel1++;
              else
              if( identi[0] == ']'  result == 0 )
                nivel1--;
              else
                {
                  ind = creaaux(identi, auxiliar, kk); kk = ind;
                }
            }
        }
      if( identi[0] != ',' )
        {
          sw++;
          switch(sw) {
          case 1 : armmov(tabars[y].res1, sennal); break;
          case 2 : armmov(tabars[y].res2, sennal); break;
          case 3 : armmov(tabars[y].res3, sennal); break;
          case 4 : armmov(tabars[y].res4, sennal); break;
          case 5 : armmov(tabars[y].res5, sennal); break;
          case 6 : armmov(tabars[y].res6, sennal); break;
          default : break;
                    }
          sennal = 0;
        }
      result = scanner(arch);
    }
  }
 else
   errrr(5);
 }
armmov(s, l)
char s[];
int l;
 {
   int swy;
   if( l == 1 )
     llenatab(s, auxiliar);
   else
   if( identi[0] == '^' )
     {
       swy = y - 1;
       switch( identi[1] ) {
```

```
        case '1' : llenatab(s, tabarg[swg].reg1); break;
        case '2' : llenatab(s, tabarg[swg].reg2); break;
        case '3' : llenatab(s, tabarg[swg].reg3); break;
        case '4' : llenatab(s, tabarg[swg].reg4); break;
        case '5' : llenatab(s, tabarg[swg].reg5); break;
        case '6' : llenatab(s, tabarg[swg].reg6); break;
        default : break;
                                }
      }
    else
      llenatab(s, identi);
  }
/*  */
/* ESTE PROGRAMA HACE LOS OPERACIONES NECESARIAS CUANDO SE TRATA DE
 */
/* UNA METAVARIABLE.
 */
/*  */
evalua(arch)
FILE *arch;
  {
    int swmet;
    swmet = revmet();
    if( swmet == 1 )
        integra(arch);
    else
      {
        llenatab(tabmet[p].nombre, identi);
        tabmet[p].numero = 0;
        ppp = p;
        p++;
        integra(arch);
      }
  }
defpush()
  {
    if( st < MAXLON )
      {
        tabstack[st].numero = tabmet[1].numero;
        st++;
      }
    else
      errrr(6);
  }
defpop()
  {
    if( st > 1 )
      {
        st--;
        tabmet[2].numero = tabstack[st].numero;
      }
    else
      errrr(7);
```

```
}
integra(arch)
FILE *arch;
{
   int token, result, sw, swr, m1, m2;
   token = 0; swr = 0;
   result = scanner(arch);
   if( identi[0] == '=' )
     {
       while( result != 4  lreg != 0 )
         {
           result = scanner(arch);
           if( result == 3 || result == 1 )
             {
               swr++;
               if( result == 3 )
                 {
                   ppp = pp;
                   if((revmet()) == 1)
                     token = tabmet[ppp].numero;
                   else
                       error(3);
                 }
               else
                 token = atoi(identi);
               tabmet[1].numero = token;
               defpush();
             }
           if( result == 0 )
             {
               switch(identi[0]) {
               case '+' : sw = 1; break;
               case '-' : sw = 2; break;
               case '*' : sw = 3; break;
               case '/' : sw = 4; break;
               default : break;
                                  }
             }
           if( swr == 2 )
             {
               defpop();
               m1 = tabmet[2].numero;
               defpop();
               m2 = tabmet[2].numero;
               switch(sw) {
               case 1 : token = m1 + m2;
                        break;
               case 2 : token = m2 - m1;
                        break;
               case 3 : token = m1 * m2;
                        break;
               case 4 : token = m2/m1;
                        break;
```

```
                default : break;
                        }
                tabmet[1].numero = token;
                defpush();
                swr = 1;
              }
          }
        defpop();
        tabmet[ppp].numero = tabmet[2].numero;
      }
  }
defesc()
  {
    int ind;
    char util[MAXLON];
    itoa(tabmet[2].numero, util);
    ind = creaaux(util, buffaux, k); k = ind;
    if( k < 7 )
      k = 7;
  }
subrrut(arch)
FILE *arch;
  {
    FILE *fopen(), *fclose();
    x = 1;
    while((novacio(tabnom[x].nombre)) != 0)
      {
        lseek(arch, tabnom[x].dir, 0);
        copyarc(arch, fp2);
        x++;
      }
  }
limpia()
  {
    clear(tabars[y].res1, MAXLON);
    clear(tabars[y].res2, MAXLON);
    clear(tabars[y].res3, MAXLON);
    clear(tabars[y].res4, MAXLON);
    clear(tabars[y].res5, MAXLON);
    clear(tabars[y].res6, MAXLON);
  }
errrr(l)
int l;
  {
    switch(l) {
    case 1 : printf("Esta mal estructurada la Macro-definicionn");
             break;
    case 2 : printf("Sobra un ENDMn");
             break;
    case 3 : printf("La Metavariable no se encuentra en la tablan");
             break;
    case 4 : printf("Revisar el arch. auxili.arc, las direccionesn");
             printf(" no coinciden con los Macrosn");
```

```
            break;
   case 5 : printf("Esta mal estructurada la Macro-llamada\n");
            break;
   case 6 : printf("En la funcion PUSH, se rebaso la long. del
stack\n");
            break;
   case 7 : printf("En la func. POP, el apuntador tomo el valor 0\n");
            break;
   case 8 : printf("La subrutina no se encuentra en la tabla\n");
            break;
   case 9 : printf("No se tecleo nada en la terminal\n");
             break;
   case 10 : printf("No existe la Bibl. de A. Numericos\n");
             break;
   case 11 : printf("No existe la Bibl. de Macros\n");
             break;
   case 12 : printf("No existe el arch. auxiliar\n");
             break;
   case 13 : printf("No existe el arch. del usuario en el
directorio\n");
             break;
   case 14 : printf("No existe el archivo auxil2\n");
             break;
   case 15 : printf("No existe el arch. salida.ftn\n");
             break;
   default : break;
              }
   exit();
 }
.NOLITERAL
```

A N E X O 2

TABLA DE MACROS, TABLA DE METAVARIABLES Y

CUERPO DE LOS MACROS

GENETI	1	4
AUX1ETI	5	3
AUX2AUX	8	3
ETIAUX	11	3
AUXAUX2	14	3
ETIAUX1	17	3
ETIAUX2	20	3
AUX3AUX	23	3
AUXAUX3	26	3
DEFMAT	29	3
DEFVEC	32	3
INICIND	35	3
INICCONS	38	3
LEEV	41	5
LEEVH	46	5
ESCV	51	5
ESCVH	56	5
ESCI	61	5
ESCF	66	5
ESCLET	71	5
LEEI	76	5
LEEI1	81	5
LEEF	86	5
REPEAT	91	5
UNTIL	96	4
WHILE	100	13
ENDW	113	6
FOR	119	5
ENDFOR	124	4
LEEDIM	128	4
ALIMMAT	132	14
LVECTOR	146	4
ESCMAT	150	14
NORMABS	164	13
DIVELEM	177	5
LIMVEC	182	5
ITERA	187	18
CONVERGE	205	6
IDENTI	211	17
MAYOR	228	7
INTERC	235	7
OPERACION	242	5
MULTMAT	247	11
DIVDIAG	258	6

```
 MAYDIAG   264     20
 DIVPIVOT  284     6
 CONLINPIV 290     8
******************
 ETI           100
 AUX            0
 AUX1           0
 AUX2           0
 AUX3           0
******************
        %MACRO GENETI()
        %ETI = %ETI + 1
        %AUX = %ETI
        %ENDM
        %MACRO AUX1ETI()
        %AUX1 = %ETI
        %ENDM
        %MACRO AUX2AUX()
        %AUX2 = %AUX
        %ENDM
        %MACRO ETIAUX()
        %ETI = %AUX
        %ENDM
        %MACRO AUXAUX2()
        %AUX = %AUX2
        %ENDM
        %MACRO ETIAUX1()
        %ETI = %AUX1
        %ENDM
        %MACRO ETIAUX2()
        %ETI = %AUX2
        %ENDM
        %MACRO AUX3AUX()
        %AUX3 = %AUX
        %ENDM
        %MACRO AUXAUX3()
        %AUX = %AUX3
        %ENDM
       %MACRO DEFMAT(~1, ~2, ~3)
       DIMENSION ~1(~2, ~3)
       %ENDM
       %MACRO DEFVEC(~1, ~2)
       DIMENSION ~1(~2)
       %ENDM
       %MACRO INICIND(~1)
         ~1 = 0
       %ENDM
       %MACRO INICCONS(~1, ~2)
         ~1 = ~2
       %ENDM
       %MACRO LEEV(~1, ~2)
       %GENETI()
        READ(5,%ESC) (~1(MXX), MXX = 1, ~2)
```

```
%ESC     FORMAT(F10.4)
        %ENDM
        %MACRO LEEVH(^1, ^2, ^3)
        %GENETI()
         READ(5, %ESC) (^1(^2, MJJ), MJJ = 1, ^3)
%ESC     FORMAT(F10.4)
        %ENDM
        %MACRO ESCV(^1, ^2)
        %GENETI()
         WRITE(5, %ESC) (^1(MXX), MXX = 1, ^2)
%ESC     FORMAT(F10.4, X)
        %ENDM
        %MACRO ESCVH(^1, ^2, ^3)
        %GENETI()
        WRITE(5, %ESC) (^1(^2, MJJ), MJJ = 1, ^3)
%ESC    FORMAT(F10.4, X)
        %ENDM
        %MACRO ESCI(^1)
        %GENETI()
        WRITE(5, %ESC) ^1
%ESC    FORMAT(I8)
        %ENDM
        %MACRO ESCF(^1)
        %GENETI()
        WRITE(5, %ESC) ^1
%ESC     FORMAT(F10.4)
        %ENDM
        %MACRO ESCLET(^1)
        %GENETI()
        WRITE(5, %ESC)
%ESC    FORMAT('^1')
        %ENDM
        %MACRO LEEI(^1, ^2, ^3, ^4)
        %GENETI()
         READ(5, %ESC) ^1, ^2, ^3, ^4
%ESC     FORMAT(4I8)
        %ENDM
        %MACRO LEEI1(^1)
        %GENETI()
         READ(5, %ESC) ^1
%ESC     FORMAT(I8)
        %ENDM
        %MACRO LEEF(^1)
        %GENETI()
        READ(5, %ESC) ^1
%ESC    FORMAT(F10.8)
        %ENDM
        %MACRO REPEAT()
        %GENETI()
        %PUSH
%ESC    CONTINUE
        %ENDM
        %MACRO UNTIL(^1)
```

```
      %POP
         IF ( .NOT. (~1)) GO TO %ESC
      %ENDM
      %MACRO WHILE(~1)
      %GENETI()
      %PUSH
%ESC  CONTINUE
      %GENETI()
         IF ( .NOT. (~1)) GO TO %ESC
      %AUX1ETI()
      %POP
      %PUSH
      %ETIAUX()
      %PUSH
      %ETIAUX1()
      %ENDM
      %MACRO ENDW()
      %POP
         GO TO %ESC
      %POP
%ESC   CONTINUE
      %ENDM
      %MACRO FOR(~1, ~2, ~3)
      %GENETI()
      %PUSH
         DO %ESC ~1 = ~2, ~3
      %ENDM
      %MACRO ENDFOR()
      %POP
%ESC   CONTINUE
      %ENDM
      %MACRO LEEDIM(~1)
      %ESCLET(¿N NUMERO DE INCREMENTOS?)
      %LEEI1(~1)
      %ENDM
      %MACRO ALIMMAT(~1, ~2, ~3)
      %FOR(MII, 1, ~2)
      %GENETI()
      %PUSH
       WRITE(5,%ESC) MII
      %LEEVH(~1, MII, ~3)
      %AUX1ETI()
      %POP
      %AUX2AUX()
      %ENDFOR()
      %AUXAUX2()
%ESC  FORMAT('ALIMENTE USTED LA HILERA', I4)
      %ETIAUX1()
      %ENDM
      %MACRO LVECTOR(~1, ~2)
      %ESCLET(¿ALIMENTE USTED EL VECTOR?)
      %LEEV(~1, ~2)
      %ENDM
```

```
      %MACRO ESCMAT(^1, ^2, ^3)
      %FOR(MII, 1, ^2)
      %GENETI()
      %PUSH
       WRITE(5, %ESC) MII
      %ESCVH(^1, MII, ^3))
      %AUX1ETI()
      %POP
      %AUX2AUX()
      %ENDFOR()
      %AUXAUX2()
%ESC  FORMAT('ESCRIBE EL VECTOR', I4)
      %ETIAUX1()
      %ENDM
      %MACRO NORMABS(^1, ^2, ^3)
      %INICCONS(IIM, 1)
      %INICCONS(JJM, 1)
      %FOR(MII, 1, ^2)
      %FOR(MJJ, 1, ^3)
      %GENETI()
       IF(ABS(^1(MII, MJJ) .LE. ^1(IIM, JJM)) GO TO %ESC
         IIM = MII
         JJM = MJJ
%ESC    CONTINUE
      %ENDFOR
      %ENDFOR
      %ENDM
      %MACRO DIVELEM(^1, ^2, ^3, ^4)
      %FOR(MJJ, 1, ^3)
          ^1(^2, MJJ) = ^1(^2, MJJ) / ^4
      %ENDFOR()
      %ENDM
      %MACRO LIMVEC(^1, ^2)
      %FOR(MII, 1, ^2)
       ^1(MII) = 0.0
      %ENDFOR()
      %ENDM
      %MACRO ITERA(^1, ^2, ^3, ^4, ^5)
      %FOR(MII, 1, ^4)
      ^2(MII) = ^1(MII, ^5)
      %FOR(MJJ, 1, ^4)
      IF((MJJ - MII) .EQ. 0) GO TO %ESC
      %GENETI()
      IF((MJJ - MII) .GT. 0) GO TO %ESC
        ^2(MII) = ^2(MII) - ^1(MII, MJJ) * ^2(MJJ)
      %AUX2AUX()
      %POP
      GO TO %ESC
      %ETIAUX()
      %PUSH
      %ETIAUX2()
%ESC  ^2(MII) = ^2(MII) - ^1(MII, MJJ) * ^3(MJJ)
      %ENDFOR()
```

```
      %ENDFOR()
      %ENDM
      %MACRO CONVERGE(~1, ~2, ~3, ~4)
        ~4 = ~4 + 1
       %FOR(I, 1, ~3)
        ~2(I) = ~1(I)
       %ENDFOR()
      %ENDM
      %MACRO IDENTI(~1, ~2)
      %FOR(MII, 1, ~2)
      %FOR(MJJ, 1, ~2)
      %GENETI()
      %PUSH
       IF((MII - MJJ) .EQ. 0) GO TO %ESC
         ~1(MII, MJJ) = 0.0
       %AUX1ETI()
         %ETI = %ETI - 1
         %AUX = %ETI
         GO TO %ESC
       %POP
%ESC    ~1(MII, MJJ) = 1.0
      %ENDFOR()
      %ENDFOR()
      %ETIAUX1()
      %ENDM
      %MACRO MAYOR(~1, ~2, ~3, ~4, ~5, ~6)
      %FOR( MII, ~4, ~5)
       IF((~2 - ABS(~1(MII, ~6))) .GE. 0) GO TO %ESC
         ~3 = MII
         ~2 = ABS(~1(MII, ~6))
      %ENDFOR()
      %ENDM
      %MACRO INTERC(~1, ~2, ~3, ~4)
      %FOR(MJJ, 1, ~3)
          ATMP = ~1(~2, MJJ)
          ~1(~2, MJJ) = ~1(~4, MJJ)
          ~1(~4, MJJ) = ATMP
      %ENDFOR()
      %ENDM
      %MACRO OPERACION(~1, ~2, ~3, ~4, ~5, ~6)
      %FOR(MJJ, ~2, ~3)
       ~1(~4, MJJ) = ~1(~4, MJJ) - ~6 * ~1(~5, MJJ)
      %ENDFOR()
      %ENDM
      %MACRO MULTMAT(~1, ~2, ~3, ~4, ~5)
      %FOR(MII, 1, ~3)
      %FOR(MKK, 1, ~5)
       C(MII, MKK) = 0.0
      %FOR(MJJ, 1, ~4)
       C(MII, MKK) = C(MII, MKK) +
     1  ~1(MII, MJJ) * ~2(MJJ, MKK)
      %ENDFOR()
      %ENDFOR()
```

```
      %ENDFOR()
      %ENDM
      %MACRO DIVDIAG(~1, ~2, ~3)
      %FOR(MII, 1, ~2)
       DIV = ~1(MII, MII)
       %DIVELEM(~1, MII, ~3, DIV)
      %ENDFOR()
      %ENDM
      %MACRO MAYDIAG(~1, ~2, ~3, ~4)
      %GENETI()
      %PUSH
       IF((~4 - ~3) .GE. 0) GO TO %ESC
          IMAX = ~4
          AMAX = ABS(~1(~4, ~4))
          KP1 = ~4 + 1
          %MAYOR(~1, AMAX, IMAX, KP1, ~3, ~4)
      %AUX1ETI()
      %POP
       IF((IMAX - ~4) .EQ. 0) GO TO %ESC
      %ETIAUX()
      %PUSH
      %ETIAUX1()
      %INTERC(~1, IMAX, ~3, ~4)
      %INTERC(~2, IMAX, ~3, ~4)
       DEL = -DEL
      %POP
%ESC  CONTINUE
      %ENDM
      %MACRO DIVPIVOT(~1, ~2, ~3, ~4, ~5)
       ~3 = ~1(~4, ~4) * ~3
       DIV = ~1(~4, ~4)
       %DIVELEM(~1, ~4, ~5, DIV)
       %DIVELEM(~2, ~4, ~5, DIV)
       %ENDM
       %MACRO CONLINPIV(~1, ~2, ~3, ~4)
       %FOR(MII, 1, ~4)
          AMULT = ~1(MII, ~3)
          IF((MII - ~3) .EQ. 0) GO TO %ESC
          %OPERACION(~1, 1, ~4, MII, ~3, AMULT)
          %OPERACION(~2, 1, ~4, MII, ~3, AMULT)
       %ENDFOR()
       %ENDM
.NOLITERAL
```

A N E X O 3

NOMBRE DIRECCION Y CLAVE DE LAS SUBRRUTINAS

```
 SDOT      12       0
 SAXPY     57        1
 SROTG     99       1
 SROT     149       1
 SCOPY    190       1
 SSWAP    234       1
 SNRM2    284       0
 SASUM    415       0
 SSCAL    455       1
 ISAMAX   492       0
***********
      REAL FUNCTION SDOT (N, SX, INCX, SY, INCY)
      INTEGER I, INCX, INCY, IX, IY, M, MP1, N
      REAL SX(1), SY(1), STEMP
C
C     CALCULA EL PRODUCTO ESCALAR W = X * Y
C     UTILIZA 'LOOPS' DESARROLLADOS PARA INCREMENTOS IGUALES A UNO.
C     SI N <= 0 LA FUNCION REGRESA INMEDIATAMENTE CON EL VALOR W = 0.
C
C
      STEMP = 0.0E0
      SDOT = 0.0E0
      IF (N .LE. 0) RETURN
      IF ((INCX .EQ. 1) .AND. (INCY .EQ. 1)) GO TO 20
C
C     CODIGO PARA INCREMENTOS IGUALES O DESIGUALES PERO
C     DISTINTOS DE 1.
C
      IX = 1
      IY = 1
      IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
      DO 10 I = 1,N
         STEMP = STEMP + SX(IX) * SY(IY)
         IX = IX + INCX
         IY = IY + INCY
10    CONTINUE
      SDOT = STEMP
      RETURN
C
C     CODIGO PARA AMBOS INCREMENTOS IGUALES A UNO.
C
C
20    M = MOD(N,5)
      IF (M .EQ. 0) GO TO 40
      DO 30 I = 1,M
         STEMP = STEMP + SX(I) * SY(I)
30    CONTINUE
```

```
      IF (N .LT. 5) GO TO 60
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
         STEMP = STEMP + SX(I)*SY(I) + SX(I+1)*SY(I+1) +
     1    SX(I+2)*SY(I+2) + SX(I+3)*SY(I+3) + SX(I+4)*SY(I+4)
   50 CONTINUE
   60 SDOT = STEMP
      RETURN
      END
      SUBROUTINE SAXPY (N, SA, SX, INCX, SY, INCY)
      INTEGER I, INCX, INCY, IX, IY, M, MP1, N
      REAL SX(1), SY(1), SA
C
C     CALCULA Y = A * X + Y
C     UTILIZA 'LOOPS' DESARROLLADOS PARA INCREMENTOS IGUALES A UNO.
C     SI A = 0.0 O SI N .LE. 0  LA SUBRUTINA REGRESA INMEDIATAMENTE
      IF (N .LE. 0) RETURN
      IF (SA .EQ. 0.0E0) RETURN
      IF ((INCX .EQ. 1) .AND. (INCY .EQ. 1)) GO TO 20
C
C     CODIGO PARA INCREMENTOS DESIGUALES O IGUALES PERO
C     DISTINTOS DE 1.
C
      IX = 1
      IY = 1
      IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
      DO 10 I = 1,N
      SY(IY) = SY(IY) + SA * SX(IX)
      IX = IX + INCX
      IY = IY + INCY
   10 CONTINUE
      RETURN
C
C     CODIGO PARA AMBOS INCREMENTOS IGUALES A 1.
C
   20 M = MOD(N,4)
      IF (M .EQ. 0) GO TO 40
      DO 30 I = 1,M
         SY(I) = SY(I) + SA * SX(I)
   30 CONTINUE
      IF (N .LT. 4) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
         SY(I) = SY(I) + SA * SX(I)
         SY(I+1) = SY(I+1) + SA * SX(I+1)
         SY(I+2) = SY(I+2) + SA * SX(I+2)
         SY(I+3) = SY(I+3) + SA * SX(I+3)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SROTG (SA, SB, C, S)
      REAL SA, SB, C, S, ROE, SCALE, R, Z
```

```
C
C     CONSTRUYE UNA ROTACION PLANA (DE GIVENS)
C     DADOS A Y B CALCULA:
C     G = SIGN(A) SI |A| > |B|     O
C     G = SIGN(B) SI |B| >= |A|
C     R = G * (A**2 + B**2)**1/2
C     C = A/R  SI  R .NE. 0   O
C     C = 1    SI   R = 0
C     S = B/R  SI  R .NE. 0   O
C     S = 0    SI   R = 0
C     LOS NUMEROS C, S, R SATISFACEN ENTONCES LA ECUACION MATRICIAL:
C
C      C   S       A      R
C                    =
C     -S   C       B      0
C
C     LA INTRODUCCION DE 'G' NO ES ESENCIAL PARA EL CALCULO DE LA
 MATRIZ DE
C     ROTACION DE GIVENS, PERO ES NECESARIA SI DESPUES SE QUIERE
 RECONSTRUIR
C     C Y S A PARTIR DE UN NUMERO. PARA ESTE PROPOSITO LA SUBRUTINA
 TAMBIEN
C     CALCULA:
C     Z = S    SI   |A| > |B|
C     Z = 1/C  SI   |B| >= |A|  C .NE. 0    O
C     Z = 1    SI   C = 0
C     LA SUBRUTINA REGRESA R EN A Y Z EN B ASI COMO C Y S.
C     SI EL USUARIO QUIERE RECONSTRUIR C Y S A PARTIR DE Z LO PUEDE
 HACES
C     DE LA SIGUIENTE FORMA:
C     SI Z = 1      DEFINA C = 0   Y   S = 1
C     Si |Z| < 1    DEFINA C = (1 - Z**2)**1/2  y  S = Z
C     Si |Z| > 1    DEFINA C = 1/Z  y  S = (1 - C**2)**1/2
      ROE = SB
      IF (ABS(SA) .GT. ABS(SB)) ROE = SA
      SCALE = ABS(SA) + ABS(SB)
      IF (SCALE .NE. 0.0E0) GO TO 10
         C = 1.0E0
         S = 0.0E0
         R = 0.0E0
         GO TO 20
10    R = SCALE * SQRT((SA/SCALE)**2 + (SB/SCALE)**2)
      R = SIGN(1.0,ROE) * R
      C = SA/R
      S = SB/R
20    Z = 1.0E0
      IF (ABS(SA) .GT. ABS(SB)) Z = S
      IF ((ABS(SB) .GE. ABS(SA)) .AND. (C .NE. 0.0E0)) Z = 1.0E0/C
      SA = R
      SB = Z
      RETURN
      END
      SUBRUTINE SROT (N, SX, INCX, SY, INCY, C, S)
```

```
      INTEGER I, INCX, INCY, IX, IY, N
      REAL SX(1), SY(1), STEMP, C, S
C
C     APLICA UNA ROTACION PLANA
C
C     X1      C   S       X1
C       =
C     Yi     -S   C       Yi     PARA i = 1,....,N
C
C     Si N <= 0 o Si C = 1  S = 0 LA SUBRUTINA REGRESA INMEDIATAMENTE.
C
      IF (N .LE. 0) RETURN
      IF ((INCX .EQ. 1) .AND. (INCY .EQ. 1)) GO TO 20
C
C     CODIGO PARA INCREMENTOS DESIGUALES O IGUALES PERO
C     DISTINTOS DE 1.
C
      IX = 1
      IY = 1
      IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
      DO 10 I = 1,N
         STEMP = C * SX(IX) + S * SY(IY)
         SY(IY) = C * SY(IY) - S * SX(IX)
         SX(IX) = STEMP
         IX = IX + INCX
         IY = IY + INCY
   10 CONTINUE
      RETURN
C
C     CODIGO PARA AMBOS INCREMENTOS IGUALES A 1
C
   20 DO 30 I = 1,N
         STEMP = C * SX(I) + S * SY(I)
         SY(I) = C * SY(I) - S * SX(I)
         SX(I) = STEMP
   30 CONTINUE
      RETURN
      END
      SUBROUTINE SCOPY (N, SX, INCX, SY, INCY)
      INTEGER I, INCX, INCY, IX, IY, M, MP1, N
      REAL SX(1), SY(1)
C
C     COPIA EL VECTOR X en el Y = X
C     Si N <= 0   LA SUBRUTINA REGRESA INMEDIATAMENTE
C
      IF (N .LE. 0) RETURN
      IF ((INCX .EQ. 1) .AND. (INCY .EQ. 1)) GO TO 20
C
C     CODIGO PARA INCREMENTOS DESIGUALES O IGUALES PERO
C     DISTINTOS DE 1.
```

```
      IX = 1
      IY = 1
      IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
      DO 10 I = 1,N
         SY(IY) = SX(IX)
         IX = IX + INCX
         IY = IY + INCY
10    CONTINUE
      RETURN
C
C     CODIGO PARA AMBOS INCREMENTOS IGUALES A 1
C
20    M = MOD(N,7)
      IF (M .EQ. 0) GO TO 40
      DO 30 I = 1,M
         SY(I) = SX(I)
30    CONTINUE
      IF (N .LT. 7) RETURN
40    MP1 = M + 1
      DO 50 I = MP1,N,7
         SY(I) = SX(I)
         SY(I+1) = SX(I+1)
         SY(I+2) = SX(I+2)
         SY(I+3) = SX(I+3)
         SY(I+4) = SX(I+4)
         SY(I+5) = SX(I+5)
         SY(I+6) = SX(I+6)
50    CONTINUE
      RETURN
      END
      SUBRUTINE SSWAP (N, SX, INCX, SY, INCY)
      INTEGER I, INCX, INCY, IX, IY, M, MP1, N
      REAL SX(1), SY(1), STEMP
C
C     INTERCAMBIA LOS VECTORES X y Y X <---> Y
C     Si N <= 0   LA SUBRUTINA REGRESA INMEDIATAMENTE
C
      IF (N .LE. 0) RETURN
      IF ((INCX .EQ. 1) .AND. (INCY .EQ. 1)) GO TO 20
C
C     CODIGO PARA INCREMENTOS DESIGUALES O IGUALES PERO
C     DISTINTOS DE 1
C
      IX = 1
      IY = 1
      IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
      DO 10 I = 1,N
         STEMP = SX(IX)
         SX(IX) = SY(IY)
         SY(IY) = STEMP
         IX = IX + INCX
```

```
          IY = IY + INCY
10      CONTINUE
        RETURN
C
C       CODIGO PARA AMBOS INCREMENTOS IGUALES A 1
C
20      M = MOD(N,3)
        IF (M .EQ. 0) GO TO 40
        DO 30 I = 1,M
           STEMP = SX(I)
           SX(I) = SY(I)
           SY(I) = STEMP
30      CONTINUE
        IF (N .LT. 3) RETURN
40      MP1 = M + 1
        DO 50 I = MP1,N,3
           STEMP = SX(I)
           SX(I) = SY(I)
           SY(I) = STEMP
           STEMP = SX(I+1)
           SX(I+1) = SY(I+1)
           SY(I+1) = STEMP
           STEMP = SX(I+2)
           SX(I+2) = SY(I+2)
           SY(I+2) = STEMP
50      CONTINUE
        RETURN
        END
        REAL FUNCTION SNRM2 (N, SX, INCX)
        INTEGER N, INCX, NEXT
        REAL SX(1), CUTLO, CUTHI, HITEST, SUM, XMAX, ZERO, ONE
        DATA ZERO, ONE /0.0E0, 1.0E0/
C
C       LONGITUD O NORMA EUCLIDIANA DE UN VECTOR ALMACENADO EN SX(),
 CUYOS ELE-
C       MENTOS ESTAN SEPARADOS POR UN INCREMENTO IGUAL A INCX.
C       SI N .LE. 0  REGRESA CON RESULTADO = 0.
C       SI N .GE. 1  ENTONCES INCX DEBE SER .GE. 1
C
C       C.L. LAWSON, 1978 JAN 08
C
C       METODO DE CUATRO PASOS QUE HACE USO DE DOS CONSTANTES
 CONSTRUIDAS INTER
C       NAMENTE QUE SON PROBABLEMENTE APLICABLES A TODAS LAS
 COMPUTADORAS.
C         CUTLO = MAXIMO DE SQRT(U/EPS) EN TODAS LAS MAQUINAS CONOCIDAS.
C         CUTHI = MINIMO DE SQRT(V)     EN TODAS LAS MAQUINAS CONOCIDAS.
C       DONDE:
C         EPS = NUMERO MAS PEQUENO TAL QUE   EPS + 1. .GT. 1.
C         U   = NUMERO POSITIVO MAS PEQUENO  (LIMITE 'UNDERFLOW')
C         V   = NUMERO MAS GRANDE            (LIMITE 'OVERFLOW')
C
C       BREVE RESUMEN DEL ALGORITMO:
```

```
C
C      PASO 1      BUSCA LAS COMPONENTES CERO.
C      IR AL PASO 2 CUANDO UNA COMPONENTE  NO ES CERO y .LE. CUTLO
C      IR AL PASO 3 CUANDO UNA COMPONENTE  ES .GT. CUTLO
C      IR AL PASO 4 CUANDO UNA COMPONENTE  ES .GE. CUTHI/M DONDE,
C      M = N PARA X() REAL y M = 2 * N PARA COMPLEX.
C
C      VALORES PARA CUTLO y CUTHI:
C      DE LOS PARAMETROS DEPENDIENTES DE LA MAQUINA LISTADOS EN EL
 DOCUMENTO
C      DE LA IMSL LAS COTAS SON:
C
C      CUTLO, S.P.      U/EPS = 2**(-102) PARA HONEYWELL.
C                       UNIVAC y DEC a 2**(-103)
C                       ASI CUTLO = 2**(-51) = 4.44089E-16
C
C      CUTHI, S.P.      V = 2**127 PARA UNIVAC, HONEYWELL y DEC.
C                       ASI CUTHI = 2**(63.5) = 1.30438E19
C
C      CUTLO, D.P.      U/EPS = 2**(-67) PARA HONEYWELL y DEC.
C                       ASI CUTLO = 2**(-33.5) = 8.2318D-11
C
C      CUTHI, D.P.      COMO EN S.P. CUTHI = 1.30438D19
C
C      DATA CUTLO, CUTHI / 8.232D-11, 1.304D19 /
C      DATA CUTLO, CUTHI / 4.441E-16, 1.304E19 /
C
       DATA CUTLO, CUTHI / 4.441E-16, 1.304E19 /
C
       IF ( N .GT. 0) GO TO 10
          SNRM2 = ZERO
          GO TO 300
C
10     ASSIGN 30 TO NEXT
       SUM = ZERO
       NN = N * INCX
C
C              'LOOP' PRINCIPAL
C
       I = 1
20     GO TO NEXT,(30,50,70,110)
30     IF (ABS(SX(I)) .GT. CUTLO) GO TO 85
          ASSIGN 50 TO NEXT
          XMAX = ZERO
C
C      PASO 1.  SUMA ES CERO
C
50     IF (SX(I) .EQ. ZERO) GO TO 200
       IF (ABS(SX(I)) .GT. CUTLO) GO TO 85
C
C               PREPARACION PARA EL PASO 2
C
       ASSIGN 70 TO NEXT
```

```
      GO TO 105
C
C               PREPARACION PARA EL PASO 4
C
100   I = J
      ASSIGN 110 TO NEXT
      SUM = (SUM / SX(I)) / SX(I)
105   XMAX = ABS(SX(I))
      GO TO 115
C
C     PASO 2.     SUMA ES PEQUENA
C                 ESCALA PARA EVITAR UN DESTRUCTIVO 'UNDERFLOW'
C
70    IF (ABS(SX(I)) .GT. CUTLO) GO TO 75
C
C                 CODOGO COMUN PARA LOS PASOS 2 y 4.
C                 EN EL PASO 4 SUMA ES GRANDE. SE CAMBIA DE ESCALA
C                 PARA EVITAR 'OVERFLOW'.
C
110    IF (ABS(SX(I) .LE. XMAX) GO TO 115
      SUM = ONE + SUM * (XMAX / SX(I))**2
      XMAX = ABS(SX(I))
      GO TO 200
C
115   SUM = SUM + (SX(I) / XMAX)**2
      GO TO 200
C
C              PREPARACION PARA EL PASO 3
C
75    SUM = (SUM * XMAX) * XMAX
C
C     PARA REAL o D.P.    HITEST = CUTHI / N
C     PARA COMPLEX        HITEST = CUTHI / (2*N)
C
85    HITEST = CUTHI / FLOAT(N)
C
C     PASO 3.    SUMA ES DE UN RANGO MEDIO (MID-RANGE). NO HAY
C                CAMBIO DE ESCALA.
C
      DO 95 J = 1,NN,INCX
      IF (ABS(SX(J)) .GE. HITEST) GO TO 100
95       SUM = SUM + SX(J)**2
      SNRM2 = SQRT(SUM)
      GO TO 300
C
200   CONTINUE
      I = I + INCX
      IF (I .LE. NN) GO TO 20
C
C                     FIN DEL 'LOOP' PRINCIPAL.
C               CALCULA LA RAIZ CUADRADA Y AJUSTA PARA CAMBIAR
C               DE ESCALA.
```

```
      SNRM2 = XMAX * SQRT(SUM)
300   CONTINUE
      RETURN
      ENI
      REAL FUNCTION SASUM (N, SX, INCX)
      INTEGER I, N, INCX, M, MP1, NINCX
      REAL SX(1), STEMP
C
C     LONGITUD o NORMA 1 DE UN VECTOR
C     TOMA LA SUMA DE LOS VALORES ABSOLUTOS DE LAS COMPONENTES.
C     UTILIZA 'LOOPS' DESARROLLADOS PARA INCREMENTOS IGUALES A UNO.
C     Si N <= 0 LA FUNCION REGRESA INMEDIATAMENTE CON EL VALOR W = 0.
C
C
      SASUM = 0.0E0
      STEMP = 0.0E0
      IF (N .LE. 0) RETURN
      IF (INCX .EQ. 1) GO TO 20
C
C     CODIGO PARA INCREMENTOS DISTINTOS DE UNO
C
      NINCX = N * INCX
      DO 10 I = 1,NINCX,INCX
      STEMP = STEMP + ABS(SX(I))
10    CONTINUE
      SASUM = STEMP
      RETURN
C
C     CODIGO PARA INCREMENTOS IGUALES A 1
C
20    M = MOD(N,6)
      IF (M .EQ. 0) GO TO 40
      DO 30 I = 1,M
         STEMP = STEMP + ABS(SX(I))
30    CONTINUE
      IF (N .LT. 6) GO TO 60
40    MP1 = M + 1
      DO 50 I = MP1,N,6
         STEMP = STEMP + ABS(SX(I)) + ABS(SX(I+1)) + ABS(SX(I+2))
     $    + ABS(SX(I+3)) + ABS(SX(I+4)) + ABS(SX(I+5))
50    CONTINUE
60    SASUM = STEMP
      RETURN
      END
       SUBRUTINE SSCAL(N, SA, SX, INCX)
       INTEGER I, INCX, M, MP1, N, NINCX
       REAL SA, SX(1)
C      CAMBIA LA ESCALA DE UN VECTOR X = A * X
C      UTILIZA 'LOOPS' DESARROLLADOS PARA INCREMENTOS IGUALES A 1.
C      SI N <= 0 LA SUBRUTINA REGRESA INMEDIATAMENTE
C
C
       IF (N .LE. 0) RETURN
```

```
      IF (INCX .EQ. 1) GO TO 20
C
C     CODIGO PARA INCREMENTOS DISTINTOS DE 1
C
      NINCX = N * INCX
      DO 10 I = 1,NINCX, INCX
        SX(I) = SA * SX(I)
10    CONTINUE
      RETURN
C
C     CODIGO PARA AMBOS INCREMENTOS IGUALES A 1
C
20    M = MOD(N,5)
      IF(M .EQ. 0) GO TO 40
      DO 30 I = 1,M
       SX(I) = SA * SX(I)
30    CONTINUE
      IF( N .LT. 5) RETURN
40    MP1 = M + 1
      DO 50 I = MP1,N,5
       SX(I) = SA* SX(I)
       SX(I+1) = SA * SX(I+1)
       SX(I+2) = SA * SX(I+2)
       SX(I+3) = SA * SX(I+3)
       SX(I+4) = SA * SX(I+4)
50    CONTINUE
      RETURN
       END
      INTEGER FUNCTION ISAMAX (N, SX, INCX)
      INTEGER N, INCX, IX, I
      REAL SX(1), SMAX
C
C     BUSCA EL INDICE DEL MAXIMO EN VALOR ABSOLUTO.
C     SI N <= 0 LA FUNCION REGRESA INMEDIATAMENTE CON I = 0
C
      ISAMAX = 0
      IF(N .LT. 1) RETURN
      ISAMAX = 1
      IF (N .EQ. 1) RETURN
      IF (INCX .EQ. 1) GO TO 20
C
C     CODIGO PARA INCREMENTOS DISTRINTOS DE 1
C
      IX = 1
      SMAX = ABS(SX(I))
      IX = IX + INCX
      DO 10 I = 2,N
      IF (ABS(SX(I)) .LE. SMAX) GO TO 5
      ISAMAX = I
      SMAX = ABS(SX(I))
5     IX = IX + INCX
10    CONTINUE
```

```
      RETURN
C
C     CODIGO PARA INCREMENTOS IGUALES A 1
C
20    SMAX = ABS(SX(I))
      DO 30 I = 2,N
        IF (ABS(SX(I)) .LE. SMAX) GO TO 30
        ISAMAX = I
        SMAX = ABS(SX(I))
30    CONTINUE
      RETURN
      END
.NOLITERAL
```