# Centro Latinoamericano de Demografía

CELADE

Documentos de Seminario



# FORTRAN IV

DS/3
100
1975

CURSO LATINOAMERICANO DE
PROCESAMIENTO DE DATOS (PED)
APLICADO A LAS CIENCIAS SOCIALES

# I N D I C E

# I. INTRODUCCION

Entre los lenguajes que se utilizan para comunicación con los computadores, uno de los más conocidos es el lenguaje FORTRAN (FORmula TRANslation). Es un lenguaje simbólico de tipo general que permite resolver con facilidad la representación de algoritmos para solución de problemas científicos en términos de instrucciones al computador.

La gran mayoría de los computadores posee una versión de FORTRAN, de tal manera que un programador puede procesar sus problemas en distintos computadores sin que ello le signifique hacer cambios notables en la estructura de sus programas. Fundamentalmente, las diferencias estarán relacionadas con los dispositivos de entrada y salida de datos y con las instrucciones respectivas. Si se han diseñado los programas en forma modular, los cambios se referirán principalmente al módulo de entrada o al módulo de salida o a ambos, dejando el resto de los módulos que son el cuerpo del programa con su estructura original.

Es importante señalar que al aprender un lenguaje de este tipo, es bastante fácil adquirir posteriormente el dominio de otros lenguajes de la misma categoría, dado que las instrucciones básicas, si no son iguales, al menos poseen la misma lógica de funcionamiento.

En relación al proceso de los programas, es necesario que se cumplan dos etapas:

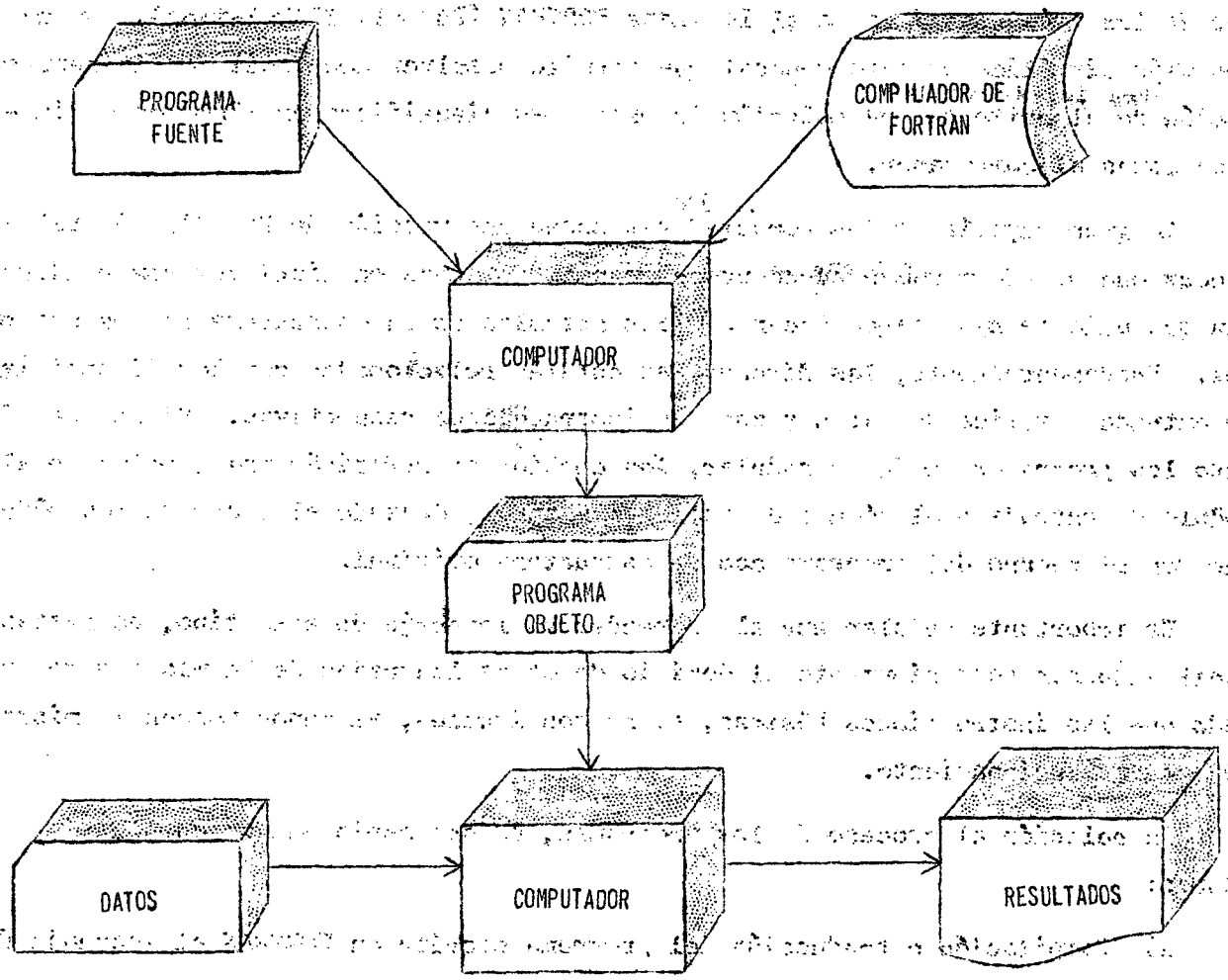
a) Compilación o traducción del programa escrito en FORTRAN al lenguaje de la máquina que se va a utilizar; y

b) Ejecución del programa traducido a lenguaje de máquina.

En la etapa de compilación, el programa escrito en FORTRAN (programa fuente) desempeña el papel de datos, los que procesará un programa escrito en lenguaje de máquina denominado COMPILADOR.

El resultado de este proceso es el mismo programa, pero ahora traducido a lenguaje de máquina (programa objeto ) y puede obtenerse en tarjetas perforadas, en cinta magnética, en disco magnético, etc. o también puede dejarse en la misma memoria de trabajo.

En la etapa de ejecución, el programa objeto procesa los datos del problema y entrega los resultados requeridos.




A continuación se describe un FORTRAN IV casi completo y en el APENDICE B se da a conocer el resto de los adelantos que tiene el lenguaje FORTRAN IV COMPLETO, aún cuando sólo están enunciados.

## II. ELEMENTOS DEL LENGUAJE

Un programa escrito en un lenguaje simbólico está compuesto por proposiciones, ordenadas secuencialmente de acuerdo con lo estipulado en el algoritmo de solución del problema. Su ejecución se efectúa en el mismo orden en que se encuentran, es decir, una a continuación de la otra.

En el lenguaje FORTRAN las proposiciones se pueden agrupar en la forma siguiente:

a) de entrada y salida
b) de asignación
c) de control
d) de especificación
e) de subprograma.

Los tipos a), b) y c) se denominan ejecutables y las de los tipos d) y e), no ejecutables.

Para que el programa pueda ser leido por el computador, se perfora normalmente en tarjetas y para ello es necesario escribir las proposiciones en hojas de codificación apropiadas.

### 1. Hoja de codificación

La hoja de codificación tiene líneas con capacidad para ochenta caracteres, lo que significa que cada línea puede ser vaciada en su totalidad en una tarjeta.

Se puede subdividir la línea en campos:

Campo 1 formado por columnas 1 a 5
Campo 2 formado por columna 6
Campo 3 formado por columnas 7 a 72
Campo 4 formado por columnas 73 a 80

a) Campo 1 (columnas 1 a 5). Se utiliza para especificar un número que identifique a la proposición. Puede variar dicho número desde 1 hasta 99999 y no puede haber dos o más proposiciones, en el mismo programa, con idéntico número.

La ubicación del número dentro del campo es arbitraria dado que una columna en blanco (Ø) no se interpreta. Se tiene así que todas las líneas siguientes

| 1 2 3 4 5 | |
|---|---|
| 1 5 | |
| 1 5 | |
| 1 5 | |
| 1 5 | |
| 1 5 | |

contienen el número 15 como identificación.

Aún cuando todas las líneas pueden llevar número de identificación, se evita hacerlo porque eso significa emplear más tiempo en la etapa de compilación. Se identifican entonces sólo aquellas líneas que van a ser referidas.

El número de identificación sólo cumple la función de identificar. Luego, su magnitud no implica prioridad en la ejecución de la proposición.

Comentarios: Parte de la documentación de un programa se puede obtener a través de explicaciones internas acerca de lo que hace el programa o una parte de él, la forma en que deben especificarse los datos y el orden en que deben ser colocados, etc. Esto se logra escribiendo la letra C en la columna 1, con lo cual el compilador no procesa la línea completa y sólo se imprime en el listado que se obtiene del programa fuente.

Ejemplo 1:

```
C EL CØMENTARIØ PERMITE
C DØCUMENTAR LØS PRØGRAMAS
C EN FØRMA INTERNA.
```

La letra O se cruza para no confundirla con el Nº0

b) Campo 2 (columna 6). Este campo contiene normalmente el carácter blanco o cero (el espacio en blanco es un carácter y como tal tiene su representación dentro del computador). Se utiliza cuando la proposición no cabe en la línea; en ese caso, se codifica cualquier carácter distinto de blanco o cero en la columna 6 de la línea siguiente para indicar que la proposición continúa en ella.

Puede haber hasta 19 líneas de continuación correspondientes a una sola proposición, pero NO PUEDE HABER MAS DE UNA PROPOSICION POR LINEA.

c) Campo 3 (columna 7 a 72). Es el que contiene la proposición propiamente tal. Con el objeto de dar más claridad a éste, es posible intercalar blancos en la medida que desee el programador. Los blancos son ignorados por el compilador excepto cuando ellos forman parte de un dato literal, en cuyo caso son considerados y tratados como blancos.

d) Campo 4 (columna 73 a 80). Este campo no es significativo para el compilador FORTRAN, por lo tanto es posible utilizarlo como parte de un comentario para identificar el programa o para verificar la secuencia de las tarjetas.

Ejemplo 2:

```
C EJEMPLO 2.
C ESTE PRØGRAMA LEE DØS
C DATØS, REALIZA UN CALCULØ,
C IMPRIME EL RESULTADØ Y
C SE DETIENE.
      READ (1,10) B,C
      A = B + C
      WRITE (3,20) A
      STØP
   10 FØRMAT(F6.2,F6.2)
   20 FØRMAT(F10.3)
      END
```

Las cuatro primeras líneas del programa anterior son de comentario. A continuación cuatro sentencias ejecutables: una de entrada (READ), una de asignación, una de salida (WRITE) y una de control (STØP). Finalmente, dos proposiciones de especificación y una de control (END) que indica el término del programa fuente.

Las proposiciones de especificación FORMAT son las únicas de especificación que pueden ir en cualquier parte del programa. Las restantes deben ir siempre al comienzo. Aquéllas indican la estructura que deben tener los datos de entrada o la forma en que se imprimirán o grabarán los resultados, de tal modo que siempre se mencionan con una o más proposiciones de entrada o salida.

## 2. Constantes

Una constante es un valor escrito en el programa fuente. Como su nombre lo indica, es un elemento que permanece fijo, invariable.

Hay cinco tipos de constantes:

ENTERAS

REALES

LOGICAS

HEXADECIMALES

LITERALES

Las reales pueden ser estipuladas con PRECISION SIMPLE o con DOBLE PRECISION.

Constante ENTERA. Es un número decimal escrito SIN punto decimal. Ocupa cuatro bytes en memoria y su magnitud máxima es:

$2^{31} - 1$, que es igual a 2147483647

Ejemplo 3:

i) Constantes enteras válidas:

0 +99999 175 -2147483647

(el signo más (+) puede ser omitido).

ii) Constantes enteras no válidas:

0 99,999 .5 2147483649

Constante REAL. Puede tener una de las tres formas siguientes:

BASICA: es un número decimal escrito CON punto decimal. Ocupa cuatro bytes en memoria. Si es positivo, el signo puede ser omitido.

BASICA seguida de EXPONENTE DECIMAL: el exponente decimal consiste de la letra E o la letra D seguida de una constante entera de uno o dos dígitos, con o sin signo. La letra E especifica SIMPLE precisión (cuatro bytes en memoria), la letra D indica DOBLE precisión (ocho bytes en memoria) y se interpretan como "diez elevado a".

Constante ENTERA seguida de EXPONENTE DECIMAL.

Magnitudes: $10^{-78}$ ($16^{-65}$) hasta $10^{75}$ ($16^{63}$)

El valor $10^{-78}$, para los efectos de cálculo, se considera equivalente a cero.
Precisión: En cuatro bytes se pueden representar seis dígitos hexadecimales (siete dígitos decimales). En ocho bytes se pueden representar catorce dígitos hexadecimales (dieciséis dígitos decimales).

Ejemplo 4:

i) Constantes reales válidas

+0.0
-100.845
9999.999
1234567.E+14 esto es 1234567. por $10^{14}$
-5.4E+02
-5.4E02 esto es -5.4 por $10^{2}$ esto es -540.
-5.4E2
-54.E1 esto es -54. por $10^{1}$
-54E1
1234567890.123456
8.7D+02
+8.7D2 esto es 8.7 por $10^{2}$ = 870.
8.7D02

ii) Constantes reales no válidas

| | |
|---|---|
| 0 | no tiene punto |
| 3,1416 | tiene coma |
| 16.28E | no tiene la constante entera del exponente |
| -15.3D-97 | excede la magnitud permitida |
| 828.524.627 | tiene más de un punto |
| -4.5D78 | excede la magnitud permitida |

Constante LOGICA. Esta constante especifica un valor lógico, "verdad" (true) o "falso" (false). Solamente hay dos constantes lógicas:

.TRUE.
.FALSE.

Cada una de estas constantes ocupa cuatro bytes en memoria. Cuando se asigna una constante lógica a una variable lógica (ver "Tipos y longitudes de variables") se está especificando que dicha variable tomará el valor TRUE o el valor FALSE.

Al escribir la constante debe ser precedida y seguida por punto.

Constante HEXADECIMAL. Es un número hexadecimal precedido por la letra Z. Un byte contiene dos dígitos hexadecimales. Si el número contiene un número impar de dígitos, se agrega un cero hexadecimal a la izquierda del número.

Si la longitud de la variable que va a contener a la constante es mayor que la necesaria, se rellena con ceros hexadecimales por la izquierda; si la longitud es menor que la necesaria, se trunca el número por la izquierda.

Constante LITERAL. Es una cadena de caracteres alfabéticos, numéricos y/o especiales que se puede expresar en las formas siguientes:

a) Encerrada entre apóstrofos

b) Precedida por wH en que w es el número de caracteres que contiene la cadena.

Cada carácter requiere un byte de almacenamiento. El número de caracteres en la cadena no puede ser mayor que 255.

```
C EJEMPLO 5.
C EN ESTE PRØGRAMA SE USAN
C CØNSTANTES ENTERAS Y
C REALES.
      READ (1,10) B,C
C SE MULTIPLICA B PØR LA
C CØNSTANTE REAL 3.1416
      A = B * 3.1416
C SE DIVIDE C PØR LA
C CONSTANTE ENTERA 2
      D = C / 2
      WRITE (3,20) A,D
      STØP
   10 FØRMAT (F6.2,F6.2)
   20 FØRMAT (F10.3,F10.3)
      END
```

## 3. Nombres simbólicos

Son identificadores formados por uno a seis caracteres alfanuméricos, esto es, alfabéticos (A a Z y $ que se incluye en este grupo) o numéricos (0 a 9). El primer carácter debe ser alfabético.

Ejemplo 6:

i) Nombres válidos

A

ZETA1

$152

NUMERO

ii) Nombres no válidos

| | |
|---|---|
| A-B | carácter extraño, el guión |
| ZETA. | carácter extraño, el punto |
| 4ALFA | empieza con carácter numérico |
| TEMP235 | tiene más de seis caracteres |

## 4. Variables

Tiene el mismo significado que en álgebra. Una variable es un símbolo que representa uno de muchos valores numéricos o uno de los dos valores lógicos.

En los problemas que se han visto, A, B, C y D son variables. Las variables se identifican con un nombre simbólico, el cual puede servir también como ayuda en la documentación del programa si es que se elige con una significación adecuada. Por ejemplo, el área de un rectángulo se puede calcular con la expresión siguiente:

S = A * B

pero es mucho más significativo escribir:

AREA = ANCHØ * LARGØ

### A. Tipos y longitudes de variables

Los tipos de variables son los mismos que los tipos de constantes, esto es:

- ENTERAS
- REALES
  - SIMPLE PRECISION
  - DOBLE PRECISION
- LOGICA

A cada tipo de variable le corresponde una longitud normal y una opcional las cuales determinan la cantidad de bytes en memoria que ocupará el valor. La longitud opcional debe ser declarada mediante una sentencia de especificación

| Tipo de variable | long. normal | long. opcional |
|---|---|---|
| Entera | 4 | 2 |
| Real | 4 | 8 |
| Lógica | 4 | 1 |

Existen tres formas que permiten declarar el tipo de una variable:

-Especificación predefinida

-Mediante proposición IMPLICIT

-Mediante proposiciones de especificación

a) Especificación predefinida

Se obtiene a través del nombre de la variable. Si el primer carácter del nombre es: I,J,K,L,M ó N, el tipo de la variable queda automáticamente definido como ENTERO; cualquier otro carácter define una variable de tipo REAL con SIMPLE PRECISION. Tanto las variables enteras como las reales ocupan cuatro bytes en memoria.

Se puede observar que con la especificación predefinida no es posible obtener variables de tipo real con doble precisión y tampoco variables lógicas. En este caso es necesario recurrir a las proposiciones de especificación explícitas.

b) Proposición IMPLICIT

Al igual que en la especificación predefinida la proposición IMPLICIT declara el tipo de la variable haciendo uso del primer carácter del nombre. Sin embargo, el programador tiene la opción de asignar un rango de caracteres alfabéticos a un tipo determinado de variable.

La proposición IMPLICIT anula los efectos de la especificación predefinida.

La estructura completa de la proposición como asimismo ejemplos de su utilización se verán en el capítulo "Proposiciones de especificación".

c) Proposiciones de especificación de tipo

Las proposiciones de especificación de tipo declaran explícitamente el tipo de una o más variables. Su efecto anula el de la proposición IMPLICIT y el de la especificación predefinida.

La estructura completa de las proposiciones como también ejemplos de la forma de utilización se verán en el capítulo "Proposiciones de especificación".

## 5. Arreglos

Si se trabaja con gran cantidad de variables, el problema mayor que se presenta en su manejo es la identificación de cada una de ellas. Junto con tener que crear el nombre es conveniente pensar en que él sea significativo y es bastante difícil, aparte de la pérdida de tiempo que ello implicaría, asignarles nombre a cada una de tres mil, cuatro mil o más variables, como sería el caso de aquellos problemas de tipo estadístico en que el volumen de información es cuantioso. Además, sería necesario repetir instrucciones que permiten efectuar una operación con un dato, para cada uno de los datos que va a ser procesado. Por ejemplo, si se va a acumular la suma de cinco datos A, B, C, D y E en un contador S, las instrucciones serán:

$$S = A$$
$$S = S + B$$
$$S = S + C$$
$$S = S + D$$
$$S = S + E$$

La notación matemática del problema es sencilla:

$$S = \sum_{i=1}^{i=5} A_i$$

o lo que es lo mismo:

$$S = A_1+A_2+A_3+A_4+A_5$$

En rigor lo que se ha hecho es sumar los elementos de un arreglo de nombre A. El nombre se hace extensible a los componentes del arreglo pero además, a cada uno de ellos se lo identifica en forma concreta con el subíndice.

El subíndice indica realmente la ubicación que tiene el elemento dentro del conjunto. Se llama subíndice para diferenciarlo del superíndice que se escribe a la derecha sobre la variable, en la misma forma que un exponente.

El conjunto o arreglo recibe en matemáticas el nombre de VECTOR LINEA o VECTOR COLUMNA, dependiendo de si está escrito horizontal o verticalmente.

En FORTRAN se utiliza el mismo criterio expuesto antes. El nombre genérico es un nombre simbólico y para hacer referencia a un elemento determinado del arreglo se indica su posición, encerrada entre paréntesis, a continuación del nombre. En el ejemplo visto es necesario un subíndice para especificar la ubicación del dato. Se trata entonces de un arreglo unidimensional o lineal.

Ejemplo 7:

Se tiene la siguiente lista de datos:

| | | | | |
|---|---|---|---|---|
| 1 | 35 | -6 | 28 | 9 |

Si el nombre genérico que se asigna a este arreglo es A, los elementos serán:

A(1) = 1

A(2) = 35

A(3) = -6

A(4) = 28

A(5) = 9

La ubicación en memoria de estos datos será uno a continuación del otro, en orden ascendente según el índice.

Debido al uso de subíndices las variables toman el nombre de VARIABLES SUBINDICADAS o variables con índice.

Considérese ahora el siguiente arreglo de datos:

Columnas

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| renglón 1 | 51 | 4 | 36 | 2 |
| renglón 2 | -5 | 13 | 24 | -1 |
| renglón 3 | 10 | 25 | -3 | -14 |

Este es un arreglo bidimensional de tres por cuatro elementos. Si a este arreglo se le llama MATRIZ (igual que la denominación que se da en matemáticas a este tipo de conjuntos), éste será el nombre genérico de los doce elementos y para referirse a alguno en particular se indicará a continuación de él, entre paréntesis, primero el renglón en el que se encuentra y después la columna correspondiente. Ambos separados entre sí por coma.

Ejemplo 8:

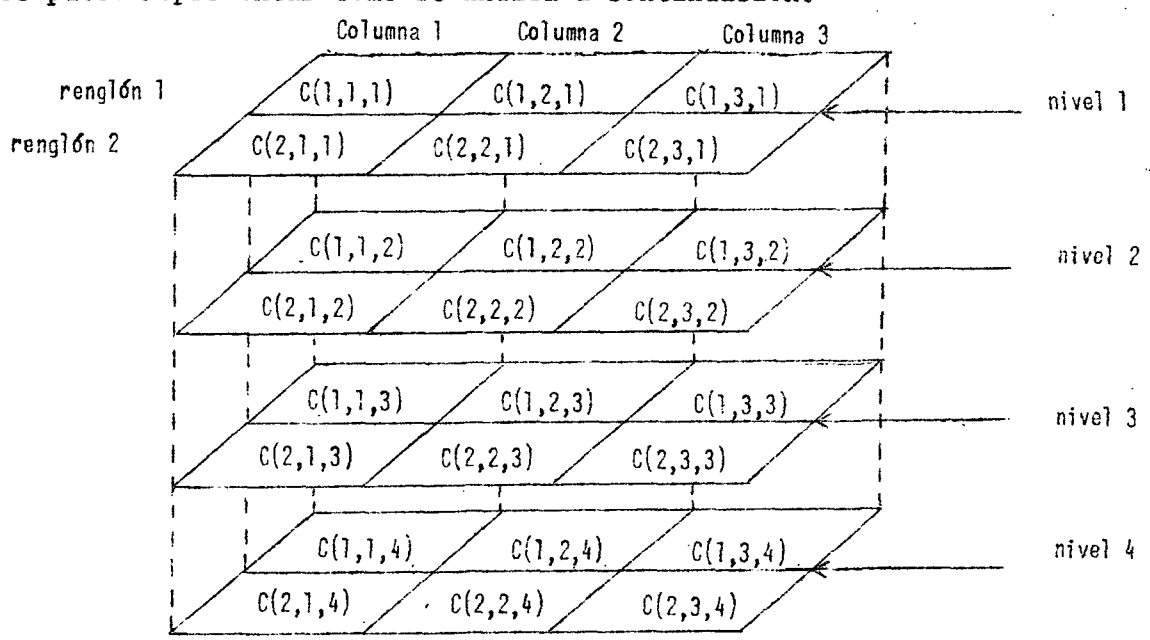
MATRIZ (3,4) es el nombre de la variable que pertenece al arreglo llamado MATRIZ y que está ubicada en el renglón 3 y en la columna 4.

Para saber la ubicación que tendrá en memoria cada uno de los elementos respecto a los demás, se hace variar "más rápido el índice de la izquierda que el de la derecha". Se tiene así, por ejemplo, en un arreglo B de tres por cuatro elementos, donde la flecha indica la secuencia de almacenamiento, que éste será:

| | | |
|---|---|---|
| B (1,1) | B(2,1) | B(3,1) |
| B(1,2) | B(2,2) | B(3,2) |
| B(1,3) | B(2,3) | B(3,3) |
| B(1,4) | B(2,4) | B(3,4) |

Nótese que este ordenamiento corresponde justamente al inverso del utilizado en álgebra.

Un arreglo tridimensional, supóngase C, de dos por tres por cuatro elementos se puede representar como se indica a continuación:



Su ubicación en memoria se obtendrá haciendo variar "rápido el índice de la izquierda, lento el del centro y más lento el de la derecha". Se obtiene entonces:

```
C(1,1,1)  C(2,1,1)---C(1,2,1) C(2,2,1)
C(1,3,1)  C(2,3,1)---C(1,1,2) C(2,1,2)
C(1,2,2)  C(2,2,2)---C(1,3,2) C(2,3,2)
C(1,1,3)  C(2,1,3)---C(1,2,3) C(2,2,3)
C(1,3,3)  C(2,3,3)---C(1,1,4) C(2,1,4)
C(1,2,4)  C(2,2,4)---C(1,3,4) C(2,3,4)
```

A. Subíndices

Se ha visto en los ejemplos de arreglos que el subíndice es un elemento que permite ubicar a un dato dentro del arreglo. Dado que la ubicación no puede ser fraccionada, el subíndice debe ser un número ENTERO.

FORTRAN NO ACEPTA SUBINDICES NEGATIVOS NI DE VALOR CERO.

FORTRAN IV Básico acepta un máximo de TRES subíndices lo que equivale a arreglos de tres dimensiones. Los subíndices pueden tener una de las siguientes SIETE formas:

a) V
b) C'
c) V + C'
d) V - C'
e) C * V
f) C * V + C'
g) C * V - C'

donde:

V es una variable entera, sin signo y sin subíndices

C y C' son constantes enteras sin signo.

Cualquiera que sea la forma de subíndice utilizada, el resultado evaluado no debe sobrepasar la dimensión correspondiente a ese subíndice.

Ejemplo 9:

i) Variables subindicadas válidas:

```
ARRAY (IHØLD)
NEXT (18)
MATRIZ (I + 2)
L (I-5)
A (6 * L)
Z (2 * J + 1)
ALFA (4 * M - 3)
```

ii) Variables subindicadas no válidas:

| | |
|---|---|
| ARRAY (-L) | la variable debe ser sin signo |
| LISTA (I-2.) | la constante debe ser entera |
| MATRIZ (-7*J) | la constante debe ser sin signo |
| W (M(3)) | la variable debe ser sin subíndices |
| NEXT (0) | el subíndice no debe ser cero |
| PAGO (J*2) | la constante debe preceder a la variable |
| TOTAL (2+K) | la variable debe preceder a la constante |

FORTRAN IV Completo acepta hasta SIETE dimensiones. En cuanto a los subíndices, pueden contener:

a) expresiones aritméticas (ver "Expresiones")
b) variables subindicadas
c) resultados reales que se convierten a enteros
d) referencias a funciones (ver "Subprogramas")

Ejemplo 10:

```
TABLA (A*2+3,B/5)
TEMP (I(5)-2,C(12,j))
BETA (A+3.8)
```

B. Reserva de Memoria para los arreglos

Para poder reservar memoria a los arreglos, sean estos de datos o de resultados, el compilador debe conocer el tipo del arreglo y la cantidad de elementos que contendrá. Esta información la obtiene de la proposición DIMENSION (ver "Proposiciones de especificación"). En ella se especifica el último elemento de cada arreglo, dado que la ubicación de él corresponde a los límites máximos de cada dimensión. El tipo del arreglo, o lo que es lo mismo de sus elementos, se obtiene en igual forma que el tipo de las variables.

El PRODUCTO DE LOS LIMITES permite obtener la cantidad de elementos del arreglo y junto con el TIPO de éste, la cantidad de memoria que es necesario reservar.

```
C EJEMPLO 11.
C USØ DE ARREGLØ
      DIMENSIØN X(5)
      READ (1,10) A,B
      X(1) = A + B
      X(2) = A - B
      X(3) = A * B
      X(4) = A / B
      X(5) = X(3) * X (2)
C SE IMPRIMEN TØDØS LØS
C ELEMENTØS DEL ARREGLØ X
      WRITE (3,20) X
      STØP
   10 FØRMAT (2F6.2)
   20 FØRMAT (5F10.3)
      END
```

## 6. Expresiones

### A. Expresión Aritmética

Una expresión aritmética se define como: una constante, variable, referencia de función (ver "Subprogramas") o combinación de ellas entre sí con operadores aritméticos.

Se puede hacer uso de paréntesis en la misma forma que en álgebra.

a) Operador aritmético.

Los operadores aritméticos son símbolos que representan operaciones que deben efectuarse entre expresiones aritméticas.

| Símbolo | Operación |
|---|---|
| ** | exponenciación |
| * | multiplicación |
| / | división |
| + | adición |
| - | substracción |

b) Normas para escribir expresiones aritméticas

Con el objeto de evitar interpretaciones erróneas de expresiones aritméticas, por ambiguedad en su escritura, es necesario cumplir las siguientes normas:

i) Toda operación entre expresiones aritméticas debe ser indicada mediante un operador.

Ejemplo: A por B debe escribirse A * B dado que AB es un nombre simbólico.

ii) No pueden aparecer dos operadores aritméticos contiguos.

Ejemplo: A por -B debe escribirse A * (-B) y no A * -B

iii) Para evaluar una expresión aritmética se procede de izquierda a derecha respetando la jerarquía siguiente:

1° Evaluación de funciones (ver "Subprogramas")
2° Exponenciación
3° Multiplicación y división
4° Adición y substracción

La excepción la constituyen operaciones de exponenciación en secuencia, en cuyo caso la evaluación se efectúa de derecha a izquierda.

Ejemplo 12:

1) A * B/C+D ** E

1° Cálculo de A * B queda X/C+D ** E

2° Cálculo de X/C queda Y+D ** E

3° Cálculo de D ** E queda Y+Z

4° Cálculo de Y+Z

2) A ** B ** C

1° Cálculo de B ** C queda A ** X

2° Cálculo de A ** X

iv) El tipo del resultado obtenido al evaluar una expresión aritmética está determinado por el tipo de las variables, constantes o resultados de funciones que conforman la expresión. La tabla que figura a continuación muestra todas las combinaciones posibles:

| + - * / ** | ENTERO | REAL, SIMPLE PRECISION | REAL, DOBLE PRECISION |
|---|---|---|---|
| ENTERO | ENTERO | RSP | RDP |
| REAL, SIMPLE PRECISION | RSP | RSP | RDP |
| REAL, DOBLE PRECISION | RDP | RDP | RDP |

B. Expresión LOGICA

Una expresión lógica se define como una constante lógica, una variable lógica, una referencia a función lógica, una expresión de relación o una combinación de ellas entre sí con operadores lógicos.

Expresión DE RELACION. Se obtiene al combinar dos expresiones aritméticas con un operador de relación.

a) Operador de relación

El operador de relación debe estar precedido y seguido por punto, separando las expresiones aritméticas. El resultado que se obtenga será siempre TRUE o FALSE.

| Operador de relación | Significado |
|---|---|
| .GT. | mayor que (>) |
| .GE. | mayor o igual que (≥) |
| .EQ. | igual a (=) |
| .LE. | menor o igual que (≤) |
| .LT. | menor que (<) |
| .NE. | no igual a (≠) |

Ejemplo 13:

Sea A = 6. K = 9 L = 3

| Expresión de relación | Valor |
|---|---|
| A.GE.8. | falso |
| K.EQ.9 | verdadero |
| L.LE.K | verdadero |
| A.NE.L | verdadero |

b) Operadores lógicos

Existen tres operadores lógicos, cada uno de los cuales debe estar precedido y seguido por punto. Solamente aquellas expresiones que al ser evaluadas tienen el valor verdadero o falso pueden combinarse con los operadores lógicos.

| Operador lógico | Uso | Significado |
|---|---|---|
| .NØT. | .NØT.A | Si A es verdad entonces la expresión tiene el valor falso; si A es falso entonces la expresión tiene el valor verdad. |
| .AND. | A.AND.B | Si A y B son verdad, entonces la expresión tiene el valor verdad. Si alguno de los dos o ambos tienen el valor falso la expresión tiene el valor falso. |
| .ØR. | A.ØR.B | Si alguno de los dos o ambos tienen el valor verdad la expresión tiene el valor verdad; si ambos tienen el valor falso, la expresión tiene el valor falso. |

Las únicas secuencias válidas de operadores lógicos son: .AND. .NØT. y .ØR. .NØT.

Ejemplo 14:

Sea I = 8 X = 55.4 e Y = 100.

| Expresión lógica | Valor |
|---|---|
| .NØT.(X.NE.Y) | falso |
| X.NE.Y.AND.X.LT.I | falso |
| Y.GT.1..ØR.X.GE.55.4 | verdad |

Evaluación de una expresión lógica: para evaluar una expresión lógica se debe respetar la jerarquía siguiente:

1° Evaluación de funciones
2° Exponenciación (**)
3° Multiplicación y división (* y /)
4° Adición y substracción (+ y -)
5° Relaciones (.GT.,.GE.,.EQ.,.LE.,.LT.,.NE.)
6° .NØT.
7° .AND.
8° .ØR.

Ejemplo 15:

1) Y.GT.A + D ** I.AND..NØT.(X.NE.Y).ØR.N

Pasos:

1° Evaluación de D ** I; queda Y .GT.A + Z.AND..NØT.(X.NE.Y).ØR.N
2° Evaluación de A + Z; queda Y.GT.W.AND..NØT.(X.NE.Y).ØR.N
3° Evaluación de X.NE.Y; queda Y.GT.W.AND..NØT.V.ØR.N
4° Evaluación de Y.GT.W; queda U.AND..NØT.V.ØR.N
5° Evaluación de .NØT.V; queda U.AND.T.ØR.N
6° Evaluación de U.AND.T; queda S.ØR.N
7° Evaluación de S.ØR.N

2) (A.AND.(B.ØR.C)).ØR.(D.AND.E)

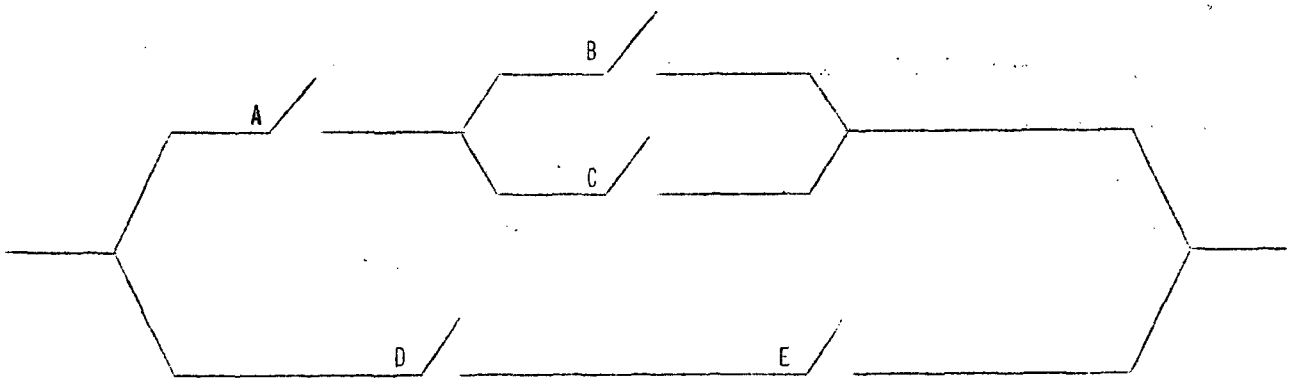
1° Evaluación de B.ØR.C; queda (A.AND.Z).ØR.(D.AND.E)

2° Evaluación de A.AND.Z; queda Y.ØR.(D.AND.E)

3° Evaluación de D.AND.E; queda Y.ØR.X

4° Evaluación de Y.ØR.X

La expresión de este ejemplo es equivalente al circuito que se indica a continuación:




## III. PROPOSICIONES

Se definen como proposiciones aquellas expresiones cuya traducción, hecha por el compilador, equivalen, en la mayoría de los casos, a varias instrucciones en lenguaje de máquina o a reservas de espacio de memoria, creación de tablas de símbolos, etc.

### 1. Proposición de asignación: aritmética y lógica

La proposición de asignación, como su nombre lo indica, permite asignar a una variable el resultado de una expresión aritmética o de una expresión lógica. Para ello se utiliza el operador de definición, que es un signo igual (=) y que se traduce como "se define por".

i) Estructura de la proposición

a = b

donde:

a: es cualquier variable con o sin subíndices

b: es cualquier expresión aritmética o lógica

ii) Función. La variable que figura al lado izquierdo del símbolo de definición se define por el valor que resulta al evaluar la expresión que está al lado derecho del símbolo. Si la variable que se está definiendo tenía algún valor, éste queda borrado por el nuevo. El tipo de la variable definida tiene prioridad sobre el tipo de resultado obtenido para la expresión, cuando ésta sea aritmética.

Si b es una expresión aritmética, a debe ser una variable real o entera. Si b es una expresión lógica, a debe ser una variable lógica.

Ejemplo 16:

Suponer que el tipo de las siguientes variables ha sido especificado como se indica a continuación:

| Variable | Tipo |
|---|---|
| A,B,C,D | Reales precisión simple |
| E,F | Reales precisión doble |
| G,H,I,J | Enteras |
| L,M | Lógicas |

De acuerdo con esas especificaciones se ilustra el funcionamiento de la proposición de asignación con los ejemplos que siguen:

| | |
|---|---|
| A = B | El valor de B define a la variable A |
| C = E*D | El valor de la expresión es de doble precisión La parte más significativa de ese valor define a C |
| F = G | El valor de la variable G se convierte a real de doble precisión y define a la variable F |
| H = D | La parte entera de la variable D se asigna a la variable H |
| I = I+1 | El valor de I es reemplazado por el valor de I más 1 |
| L = .FALSE. | El valor de L es reemplazado por FALSE |
| M = 3..NE.C | Si la constante real 3. no es igual al valor de la variable C, se asigna a M el valor TRUE; en caso contrario se le asigna FALSE |

A. Problemas propuestos

a) Escribir las proposiciones de asignación aritmética que corresponden a las siguientes fórmulas originales:

i) $\pi = 3{,}14159$

ii) $p = \gamma . Z + pa$

iii) $F = \frac{\gamma}{2} (bH^2 - bh^2)$

iv) $e = \dfrac{500H}{\mu \left( \frac{\delta}{2} + 500 \right)}$

v) $Q = 0{,}785D^2 . 0{,}82 \cdot 2gH^{\frac{1}{2}}$

vi) $R = \dfrac{4\alpha^2 \rho \; L}{\pi}$

vii) $t_3 = t_4 + \dfrac{\frac{x_3}{k_3} (t_1 - t_4)}{\frac{x_1}{k_1} + \frac{x_2}{k_2} + \frac{x_3}{k_3}}$

viii) $M = \dfrac{Ebh^3T}{6{,}6DN}$

ix) $P_5 = a_o x^5 + a_1 x^4 + a_2 x^3 + a_3 x^2 + a_4 x + a_5$

x) $c = \dfrac{U^2 + V^2}{U^2 - V^2}$

b) Suponiendo que se tienen definidas las variables A,B,C,D,I y J con los siguientes valores:

A=1. B=1.5 C=5. D=3. I=2 J=3

indicar cuál es el valor que se obtiene en las proposiciones aritméticas que figuran a continuación:

i) M = I/J

ii) N = J/I

```
iii)  BO = -1./(2.*C)+D**I/(4.*A**J)
 iv)   Y = (A*1.E-6+B*D**J)**(I/J)
  v)   Y = A+B/C-D**J
 vi)  CE = 1.112*D*B*C/(D-C)
vii)ALFA = C**I/B*(D**I)
viii)BETA = (-C-D)**I-(C-D)**(I+J)
 ix)  PX = A*B**J+C*B**I+D*B+A*C*D
  x)   L = (B+2*B)**((J+I)/I)
```

## 2. Proposiciones de Control

Se entiende por proposiciones de control aquéllas que permiten alterar la secuencia normal de ejecución de un programa.

### A. Proposición GO TO

Esta proposición permite transferir el control de la ejecución a otra proposición ubicada antes o después de ella. Esto se conoce como BIFURCACION y más comúnmente como SALTO dentro del programa; SALTO hacia ADELANTE o SALTO hacia ATRAS.

Existen tres clases de GO TO:

a) GO TO Incondicional

Es el que permite realizar el salto sin que haya una condición previa para su ejecución, esto es, el salto se efectúa siempre.

i) Estructura de la proposición

```
GO  TO  X
```

donde:

X: es el número de identificación de una proposición ejecutable.

ii) Función. Produce una bifurcación en el programa, de tal manera que la próxima proposición a ejecutar será aquélla que tiene el número X.

Ejemplo 17:

Calcular la suma de los enteros mayores que cero

```
C EJEMPLØ 17.
C GRUPØ DE PRØPØSICIØNES QUE
```

(Continúa)

(Continuación)

```
C PERMITE CALCULAR LA SUMA DE
C LØS ENTERØS MAYØRES QUE CERØ
      NS = 0
      I  = 1
   10 NS = NS + I
      I  = I + 1
      GØ TØ 10
```

Se puede observar que este grupo de proposiciones constituye un programa que, teóricamente, se ejecutará en forma indefinida dado que no hay un elemento que permita romper el ciclo obligado por la proposición GO TO

```
C EJEMPLØ 18.
C PRØBLEMA QUE SE PUEDE
C PRESENTAR EN EL USØ DE GØ TØ
      NS = 0
      I  = 1
      GØ TØ 5
      I = I+ 1
    5 NS = NS + I
      STØP
      END
```

El problema que muestra el grupo de proposiciones del ejemplo 18 es el que se deriva de tener una proposición (puede ser un grupo) sin identificación inmediatamente después del GO TO, lo que significa que ella nunca será ejecutada.

b) <u>GO TO computado</u>

Es el que permite realizar el salto de acuerdo con el valor que tenga en ese momento una variable determinada, esto es, hay un cierto control sobre la ejecución de la proposición. Sin embargo, pueden presentarse los mismos problemas que se vieron en el GO TO incondicional: ciclo indefinido o grupo de proposiciones que no se ejecuta.

i) Estructura de la proposición

GO TO $(x_1, x_2, \ldots, x_n), i$

donde:

los $x_i$: son números de identificación de proposiciones ejecutables, i es una variable entera, sin subíndices que debe cumplir con $1 \leq i \leq n$

ii) Función. El salto o bifurcación se realiza a la proposición cuyo número de identificación está en el lugar indicado por la variable i. Si el valor de i está fuera del rango permitido, se ejecuta la proposición que figura a continuación del GO TO (algunos compiladores indican error).

Ejemplo 19:

```
C EJEMPLØ 19.
C USØ DE GØ TØ EN
C SUMA REBUSCADA DE ENTERØS
      M = 0
      NS = 0
      I = 1
    5 M = M + NS/100
      GØ TØ (20,10),M
   20 NS = NS + I
      I = I + 1
      GØ TØ 5
   10 WRITE (3,51)NS
      STØP
   51 FØRMAT (I6)
      END
```

En el ejemplo 19 se hace uso de la característica del GO TO Computado, de tomar la proposición siguiente cuando la variable está fuera de rango (eso ocurrirá cuando M tenga valor 0). Cuando M tenga valor 1, el salto se realiza a la proposición 20 y cuando M sea 2 el salto se efectúa a la proposición 10 en que se imprime el valor obtenido por NS. ¿Cuántos enteros se suman? Conviene tener cuidado al hacer uso de la característica mencionada, dado que algunos compiladores, como se indicó antes, acusan error y otros producen resultados imprevisibles sin dar ninguna indicación.

c) GO TO asignado

Similar a la proposición anterior, en ésta se asigna un valor a la variable que indicará el lugar o meta del salto mediante una proposición ASSIGN.

i) Estructura de la proposición

ASSIGN i TO m

GO TO m,($x_1$,$x_2$,...,$x_n$)

donde:

los $x_i$ son números de identificación de proposiciones ejecutables, i es un número de identificación de proposición y debe corresponder a uno de los x, m es una variable entera, sin subíndices, a la cual se le asigna el valor de i.

ii) Función. El salto o bifurcación se realiza a la proposición cuyo número de identificación se le haya asignado a la variable m. Este número debe estar entre los $x_i$.

Ejemplo 20:

```
C EJEMPLØ 20.
C USØ DE ASSIGN Y
C GØ TØ ASIGNADØ
      ASSIGN 20 TØ L
      READ (1,51)B,C
    5 GØ TØ L,(20,30,10)
   20 A = B + C
      ASSIGN 10 TØ L
      GØ TØ 5
   30 A = B * C - 5.4
      ASSIGN 10 TØ L
      GØ TØ 5
   10 Z = A ** 2
      WRITE (3,52) Z
      STØP
   51 FØRMAT (F6.2,F6.2)
   52 FØRMAT (F10.3)
      END
```

En este caso se trata de realizar uno de dos procesos de cálculo de acuerdo con el valor asignado a la variable L con la primera proposición ASSIGN. Un proceso corresponde a las proposiciones 20 y 10, que son las que se ejecutan en el ejemplo y el otro a las proposiciones 30 y 10. La primera proposición ASSIGN puede ser colocada en el momento de ejecución del programa con el valor 20 o con el valor 30, según sea el proceso que se desea efectuar.

B. Proposición IF

Esta proposición permite efectuar un salto, de acuerdo con un resultado aritmético o lógico obtenido. Existen, por lo tanto, dos tipos de IF:

a) IF aritmético

El salto se realiza de acuerdo con el resultado obtenido al evaluar una expresión aritmética.

i) Estructura de la proposición

IF (a) $x_1,x_2,x_3$

donde:

$x_1,x_2,x_3$: son números de identificación de proposiciones ejecutables y

a : es una expresión aritmética.

ii) Función. El salto o bifurcación se realiza a la proposición $x_1$ cuando el valor de a es menor que cero, a $x_2$ si es igual a cero y a $x_3$ si es mayor que cero. Se pueden realizar las siguientes combinaciones:

$x_1=x_2$ salto si el valor de a es menor o igual a 0

$x_2=x_3$ salto si el valor de a es mayor o igual a 0

$x_1=x_3$ salto si el valor de a es distinto de 0

Ejemplo 21:

Programar el cálculo de $\sqrt{1000}$ utilizando la fórmula de aproximación de Newton:

$$y_n = \left(\frac{1}{2}\ (y_a + \frac{N}{y_a}\right)$$

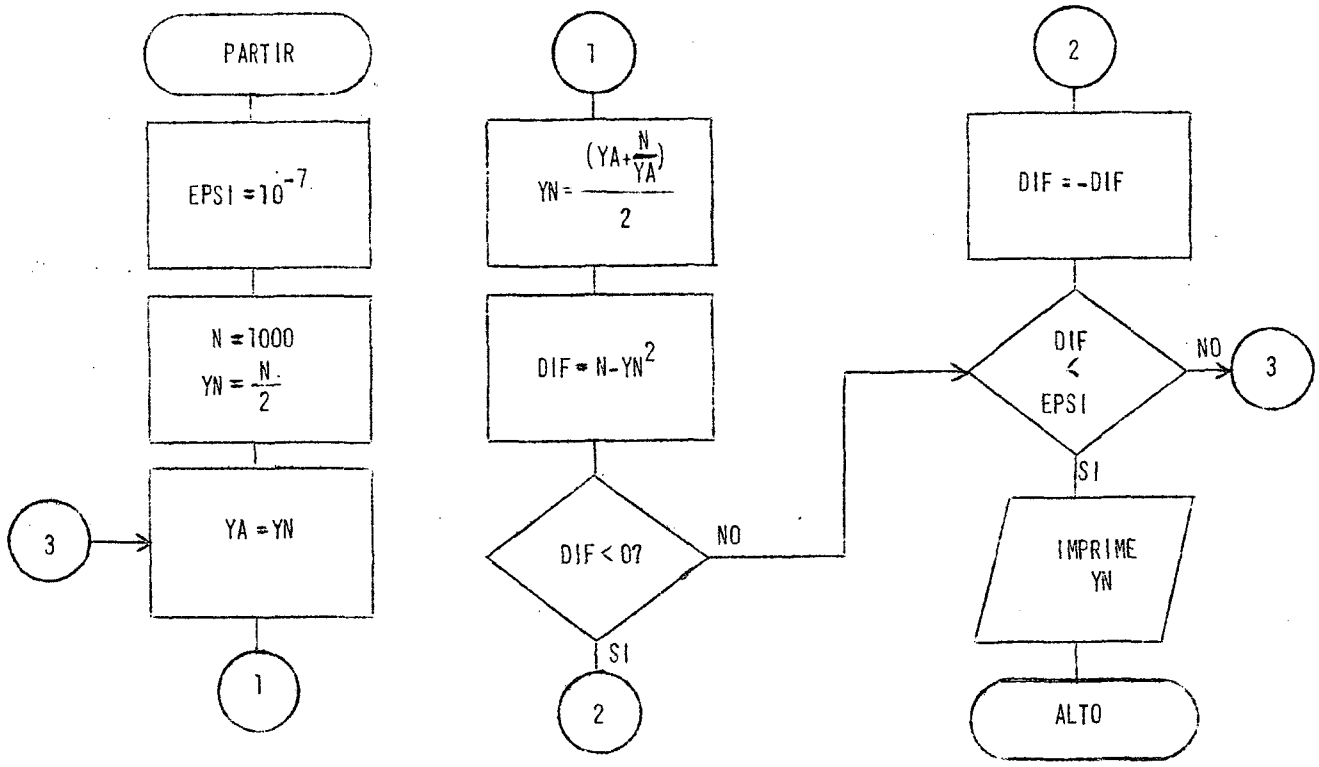
en que:

N : es el número cuya raíz se desea

$y_a$: es la raíz última obtenida (o el valor de partida)

$y_n$: es la nueva raíz

El error de aproximación debe ser menor que $10^{-7}$




```
C EJEMPLØ 21.
C USØ DE IF ARITMETICØ.
C SE CALCULA LA RAIZ CUADRADA DE N
C CØN LA FØRMULA DE NEWTØN
      EPSI = 1.E-7
      ZN = 1000.
      YN = ZN/2.
    1 YA = YN
      YN = (YA + N/YA)/2.
      DIF = ZN - YN * YN
      IF (DIF) 2,3,3
    2 DIF = -DIF
    3 IF (DIF - EPSI) 4,1,1
    4 WRITE (3,51) YN
      STØP
   51 FØRMAT (F10.3)
      END
```

Observaciones:

i) Si se cambia la proposición ZN=1000. por una proposición que lea el número, el programa queda más general.

ii) En la primera proposición IF, si el valor de DIF es igual a 0, se puede saltar a imprimir inmediatamente.

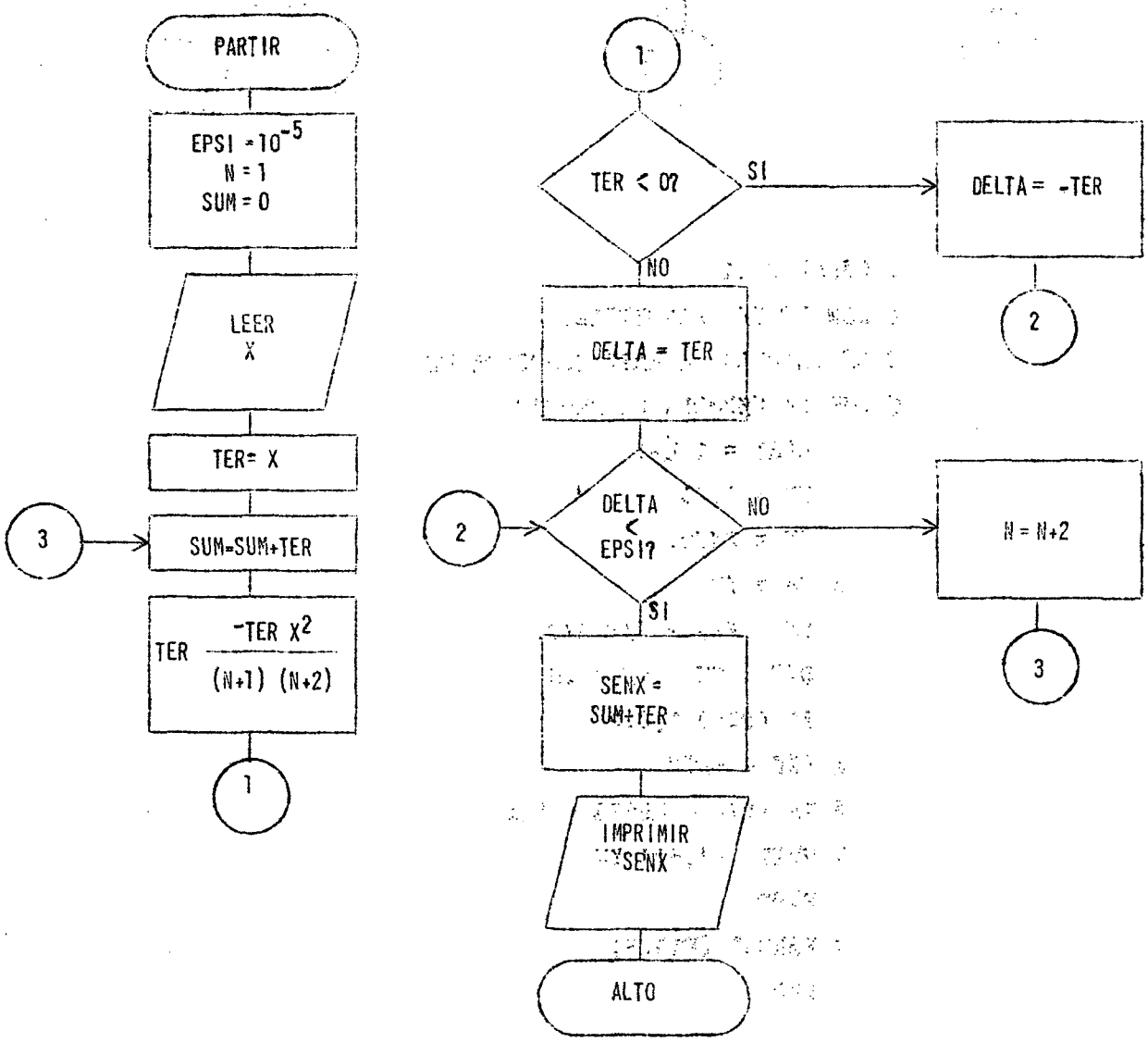
iii) La segunda proposición IF contiene en forma indirecta la comparación entre DIF y EPSI

Ejemplo 22:

Programar el cálculo de:

$$\text{senx}=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}\pm\ldots$$

Detener el cálculo cuando el último término sea, en valor absoluto, menor que $10^{-5}$.



```
C EJEMPLØ 22.
C USØ DE IF ARITMETICØ.
C SE CALCULA SEN(X) UTILIZANDØ UNA
C SERIE
      EPSI = 1.E-5
      ZN = 1.
      SUM = 0.
      READ (1,51) X
      TER = X
    6 SUM = SUM + TER
      TER = -TER*X*X/(ZN+1)/(ZN+2)
      IF (TER) 1,2,2
    1 DELTA = -TER
      GØ TØ 3
    2 DELTA = TER
    3 IF (DELTA - EPSI) 4,5,5
    5 ZN = ZN + 2.
      GØ TØ 6
    4 SENX = SUM + TER
      WRITE (3,52) SENX
      STØP
   51 FØRMAT (F6.2)
   52 FØRMAT (F10.3)
      END
```

b) <u>IF lógico</u>

El salto se realiza de acuerdo con el resultado obtenido al evaluar una expresión lógica.

i) Estructura de la proposición

IF(a)s

donde:

a: es una expresión lógica

s: es cualquier proposición ejecutable excepto una proposición DO u otra proposición IF lógica.

ii) Función. Se ejecuta la proposición s si el resultado de la expresión lógica es verdadero (TRUE); por el contrario, si es falso (FALSE) se ejecuta la proposición que sigue en secuencia al IF.

Ejemplo 23:

Hacer un programa que lea dos valores B y X. Calcular con dichos valores $A=B^2+\frac{x^2}{2}$

si se cumple que:

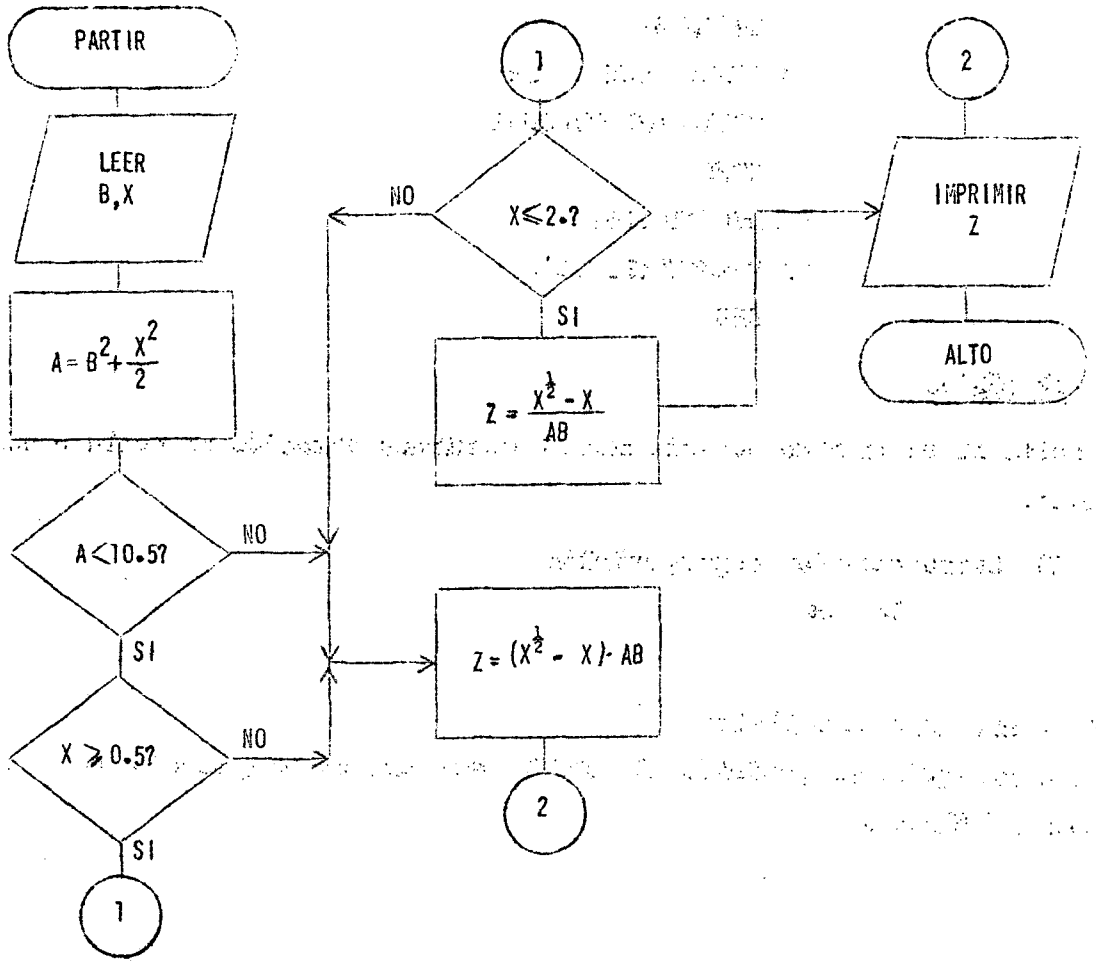
$$A < 10.5 \text{ y } 0.5 \leq x \leq 2.$$

calcular

$$Z = \frac{x^{\frac{1}{2}} - x}{AB}$$

en caso contrario, calcular

$$Z = AB(x^{\frac{1}{2}} - x)$$



```
C EJEMPLØ 23.
C USØ DE IF LØGICØ
      READ (1,51) B,X
      A = B ** 2 + X * X / 2.
      IF(A.LT.10.5.AND.(X.GE..5.AND.X.LE.2.))GØ TØ 1
      Z = A * B *(X ** .5 - X)
      GØ TØ 2
    1 Z = (X ** .5 - X)/(A * B)
    2 WRITE (3,52) Z
      STØP
   51 FØRMAT (F6.2,F6.2)
   52 FØRMAT (F10,3)
      END
```

Sugerencia: Resolver el mismo problema utilizando IF aritmêtico.

C. Proposición DO

Esta proposición permite ejecutar repetidamente, por un número de veces fijo, una o más proposiciones que conforman el "ciclo DO".

i) Estructura de la proposición

DO x i = $m_1$, $m_2$, $m_3$

donde:

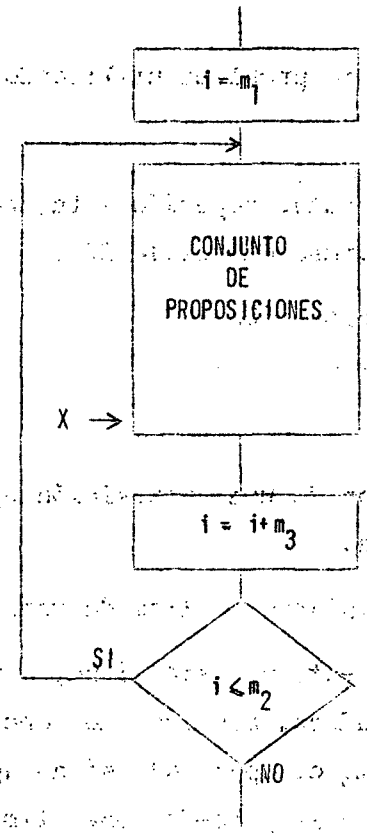
x: es el número de identificación de una proposición ejecutable que aparece después del DO en el programa.

i: es una variable entera sin subíndices llamada variable del DO

$m_1$,$m_2$,$m_3$: pueden ser constantes enteras, sin signo, mayores que cero, o variables enteras, sin signo, sin subíndice, mayores que cero. El valor de $m_2$ no puede exceder de $2^{31}$-1. El valor $m_3$ es opcional, si no aparece, se subentiende que es 1, en este caso la coma que le precede debe eliminarse.

ii) Función. En el momento de ejecutarse la proposición DO se asigna a la variable i el valor inicial $m_1$, enseguida, se ejecuta el grupo de proposiciones hasta aquella que tiene el número de identificación x. Finalizada la ejecución se suma a la variable i el incremento $m_3$ y el resultado es comparado con el valor final $m_2$. Si es menor o igual que él, se ejecuta nuevamente el grupo de proposiciones, en cambio, si el valor de la variable i excede al valor de comparación, el proceso continúa en la proposición siguiente a la que tiene el número de identificación x.

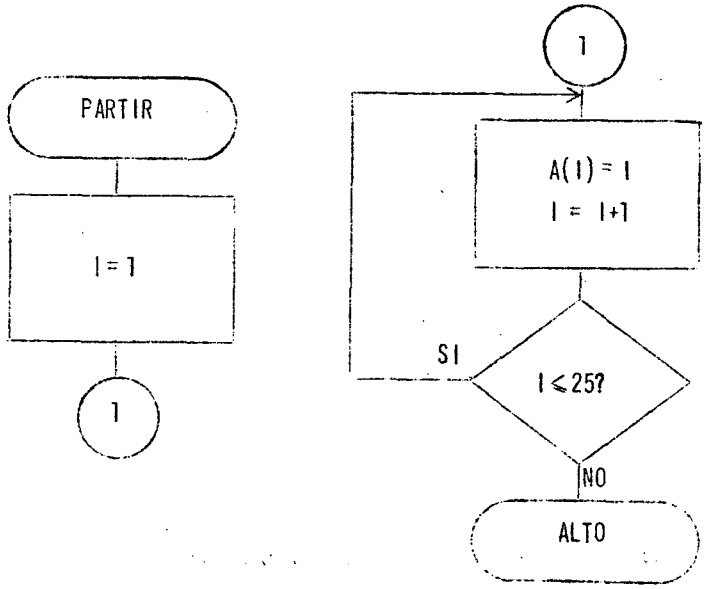
Aún cuando ocurra que $m_2$ sea menor que $m_1$ las proposiciones que forman el ciclo serán ejecutadas una vez, puesto que la comparación se realiza al término del ciclo. El diagrama de flujo que figura a continuación, corresponde al funcionamiento de la proposición DO.




Se dice que las proposiciones que forman el ciclo están en el rango del ciclo DO.

Ejemplo 24:

Formar un arreglo A con 25 elementos de tal manera que cada elemento contenga el número real equivalente a su número de orden, esto es, A(I)=I




```
C EJEMPLØ 24.
C SØLUCIØN UTILIZANDØ IF
      DIMENSIØN A(25)
      I = 1
    1 A(I) = I
      I = I + 1
      IF (I.LE.25)GØ TØ 1
      STØP
      END
C SØLUCIØN UTILIZANDØ DØ
      DIMENSIØN A(25)
      DØ 1 I = 1,25
    1 A(I) = I
      STØP
      END
```

Ejemplo 25:

En el mismo programa para el problema del ejemplo 24, calcular el producto de todos los elementos impares.

```
C EJEMPLØ 25.
C UTILIZACIØN DE DØ
      DIMENSIØN A(25)
      DØ 1 I = 1,25
    1 A(I) = I
      PA = 1.
      DØ 2 I = 1,25,2
    2 PA = PA * A(I)
      STØP
      END
```

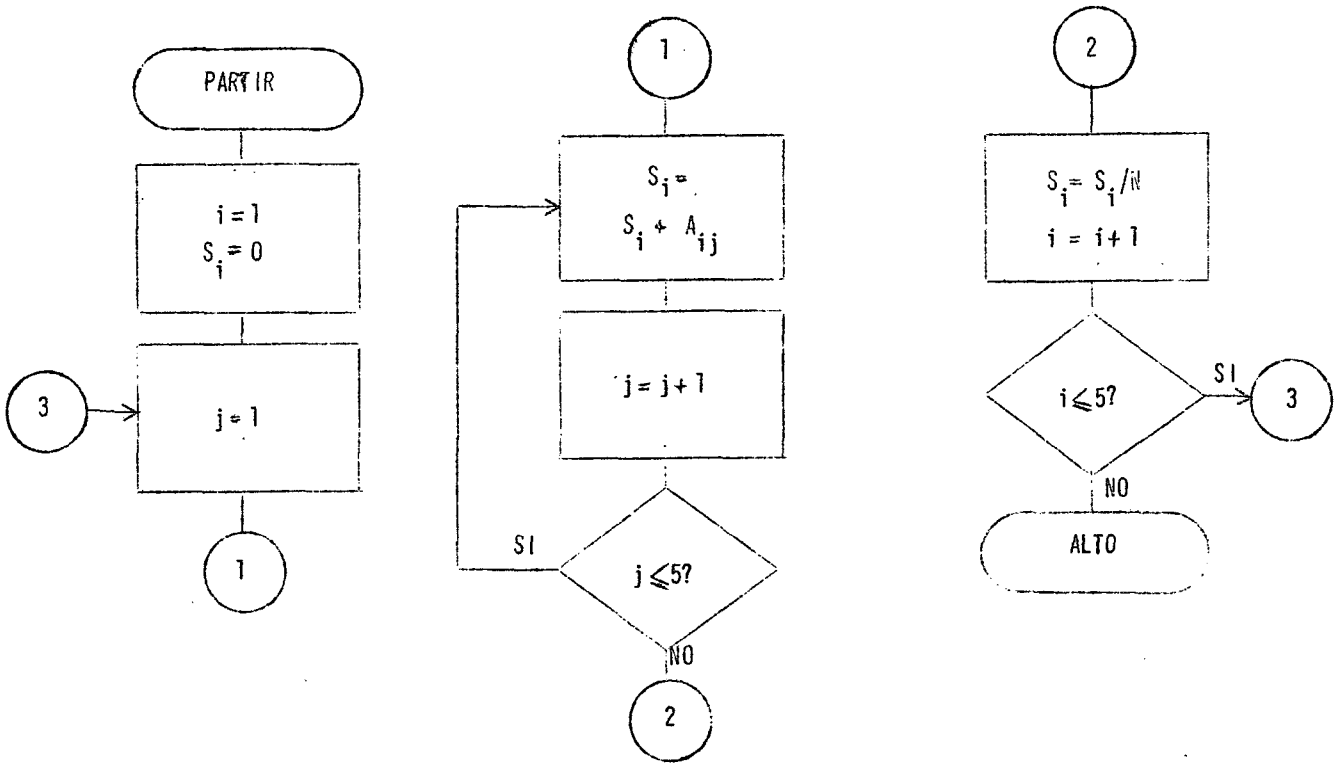
iii) Normas que deben observarse al utilizar la proposición DO

1) Los parámetros de la proposición DO, esto es, $i, m_1$, $m_2$ y $m_3$, no pueden ser cambiados por ninguna proposición que esté en el rango del ciclo DO.

2) Puede haber proposiciones DO en el rango del ciclo DO, formando así un nido de DO. Para que esta estructura sea válida, todas las proposiciones del DO interior deben estar en el rango del DO que lo contiene.

Ejemplo 26:

Se tiene un arreglo de cinco por cinco elementos. Se pide programar el cálculo del promedio de los elementos de cada renglón




```
C EJEMPLØ 26.
C APLICACIØN DE UN NIDØ DE DØ
      DIMENSIØN A(5,5),S(5)
      DØ 15 I = 1,5
      S(I) = 0.                          RANGO DEL
      DØ 10 J = 1,5          RANGO DEL   DO EXTERNO
   10 S(I) = S(I) + A(I,J)   DO INTERNO
   15 S(I) = S(I) / 5.
      STØP
      END
```

Ejemplo 27:

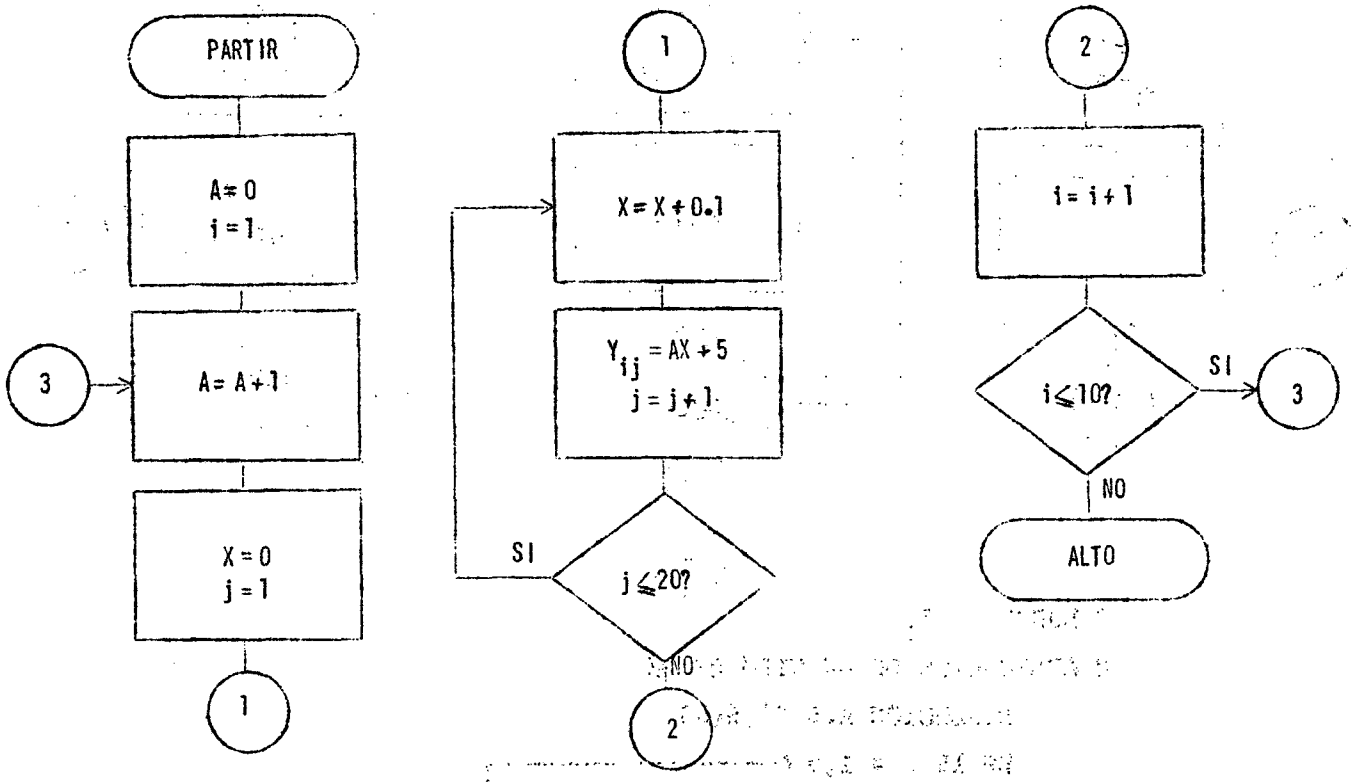
Programar el cálculo de: Y = ax + 5. para

VALOR FINAL

VALOR INICIAL — VALOR DE INCREMENTO

a = 1. (10.) 1.

x = .1 (2.) .1




```
C EJEMPLØ 27.
C APLICACIØN DE UN NIDØ DE DØ
      DIMENSIØN Y(10,20)
      A = 0.
      DØ 10 I = 1,10
      A = A + 1.
      X = 0.
      DØ 10 J = 1,20
      X = X + .1
   10 Y(I,J) = A * X + 5.
      STØP
      END
```

RANGO DEL DO INTERNO

RANGO DEL DO EXTERNO

3) Es permitido el salto desde el interior de un ciclo DO.

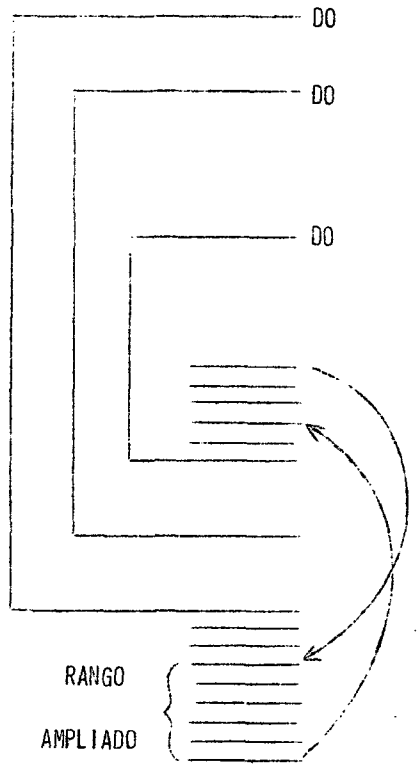
4) Se entiende como rango ampliado de un DO al conjunto de proposiciones comprendidas entre el punto meta de salto, para un salto efectuado desde el interior de un ciclo DO, y el punto donde se ordena el retorno a dicho ciclo, ambos puntos incluídos. Deben cumplirse las siguientes reglas:

El salto al interior de un ciclo DO se puede efectuar solamente si es desde un rango ampliado del DO.

El rango ampliado de un DO no debe contener otra proposición DO que también tenga rango ampliado, si este último está en el mismo módulo de programa que el primer DO.

Los parámetros de la proposición DO, esto es, i, $m_1$, $m_2$, $m_3$, no pueden ser cambiados en el rango ampliado del DO.

Ejemplo 28:



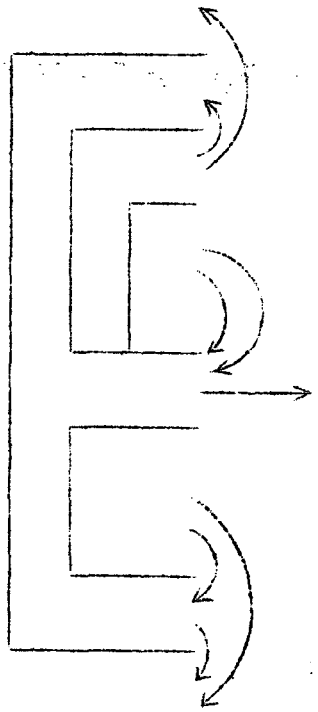

5) El número de identificación de una proposición que figura como la última en el rango de más de un DO (ejemplo 27) no puede ser utilizado como meta de salto en ningún GO TO o IF aritmético que no estén en el rango del DO más interno.

6) La última proposición en el rango de un ciclo DO no puede ser una proposición GO TO, PAUSE, STOP, RETURN, IF aritmético, DO o un IF lógico que contenga a alguna de ellas.

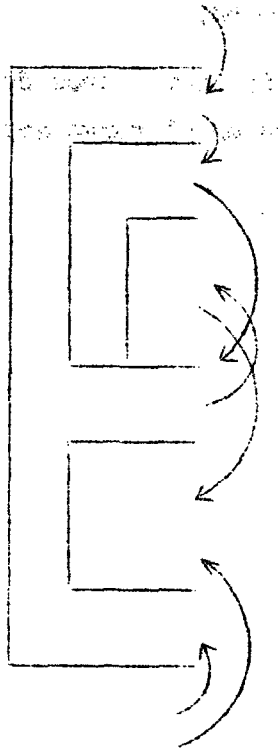
7) Se puede saltar a un subprograma desde el interior de un ciclo DO, como asimismo, retornar del subprograma al ciclo.

A continuación se indican gráficamente ejemplos de saltos permitidos y no permitidos.

Saltos permitidos



Saltos no permitidos



D. Proposición CONTINUE

Esta proposición puede ser colocada en cualquier parte de un programa, por supuesto, en un lugar que corresponda a una proposición ejecutable, sin que la secuencia de ejecución se vea afectada.

i) Estructura de la proposición

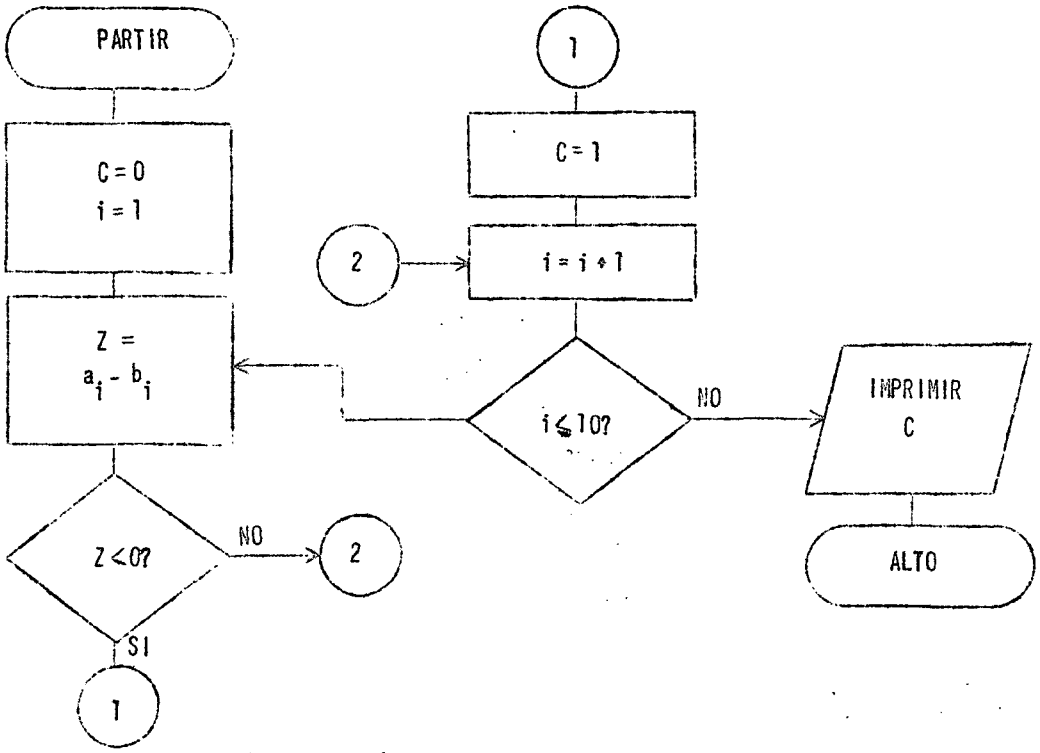
CONTINUE

ii) Función. Es una proposición "vacía" o de no-operación. Su utilidad se manifiesta al ser colocada como última proposición en un ciclo DO en aquellos casos en que éste termina con proposición GO TO, PAUSE, STOP, RETURN, IF aritmético, DO o IF lógico que contenga a alguna de ellas.

Ejemplo 29:

Se tienen dos listas, A y B, de diez elementos cada una. Se pide averiguar si alguna diferencia $A_i - B_i$ es menor que cero. Si así ocurre, hacer C=1, en caso contrario, hacer C=0.

En la solución de este tipo de problemas es necesario tener en cuenta que el proceso del ciclo DO siempre debe pasar por la última proposición de dicho ciclo, dado que inmediatamente, a continuación de la ejecución de ella, se produce el incremento de la variable del DO y su comparación con el valor final.




```
C EJEMPLØ 29.
C USØ DE LA PRØPØSICIØN CØNTINUE
      DIMENSIØN A(10),B(10)
      C = 0.
      DØ 1 I = 1,10
      IF (A(I) - B(I)) 2,1,1
    2 C = 1.
    1 CONTINUE
      WRITE (3,51) C
      STØP
   51 FØRMAT (F10.3)
      END
```

Observación: En el programa se ha eliminado el paso $Z=a_i-b_i$ porque se puede colocar directamente en la proposición IF.

Ejemplo 30:

Considerando las dos listas del problema anterior, para cada diferencia negativa de $a_i-b_i$, incrementar el primero en 2.5 y decrementar el segundo en 2.5, luego acumular la sumatoria del producto de ambos y volver a verificar la diferencia de los mismos elementos, esto es, para el mismo i.

```
C EJEMPLØ 30.
C USØ DE LA PRØPØSICIØN CØNTINUE
      DIMENSIØN A(10),B(10)
      S = 0.
      DØ 1 I = 1,10
    2 IF (A(I) - B(I)) 3,1,1
    3 A(I) = A(I) + 2.5
      B(I) = B(I) - 2.5
      S = S + A(I) * B(I)
      GØ TØ 2
    1 CØNTINUE
      WRITE (3,51) S
      STØP
   51 FØRMAT (F10.3)
      END
```

E. <u>Proposición PAUSE</u>

Se utiliza para producir una detención momentánea del proceso con el objeto de permitir alguna intervención del operador.

i) Estructura de la proposición. Tiene tres formatos:

```
PAUSE
PAUSE n
PAUSE 'mensaje'
```

donde:

n: es una constante entera sin signo de hasta cinco dígitos

'mensaje': es una constante literal

ii) Función. Junto con producirse una detención momentánea del proceso, se imprime en la máquina de escribir de la consola PAUSE 0, PAUSE y el valor de n o PAUSE y el mensaje, de acuerdo con el formato que ha sido utilizado. Una vez que se termina la acción planificada durante la pausa (análisis de resultados intermedios, cambio de formulario, montaje de discos o cintas magnéticas, etc.), se puede ordenar que continúe el proceso. En este caso la reiniciación se efectúa en la proposición siguiente a PAUSE.

F. Proposición STOP

Se utiliza para producir la detención definitiva del proceso.

i) Estructura de la proposición. Tiene dos formatos

STOP

STOP n

donde:

n: es una constante entera sin signo de hasta cinco dígitos

ii) Función. Se produce la detención definitiva del proceso y si se ha utilizado el segundo formato se imprime además STOP y el valor de n en la máquina de escribir de la consola. El segundo formato se emplea cuando se desea detectar el recorrido que ha tenido el proceso hasta llegar a obtener un resultado determinado. En este caso, se distribuyen proposiciones STOP con distintos valores de n en los puntos de término del programa que se desea controlar, de tal manera que al detenerse el proceso e imprimirse STOP y el valor de n, se ubica inmediatamente el punto de detención.

G. Proposición END

Se utiliza con el objeto de indicar el término físico de un programa fuente, sea éste programa principal o subprograma.

i) Estructura de la proposición

END

ii) Función. Indica al compilador el término de las proposiciones, por tanto, no es una proposición que produzca detención del proceso para lo cual está destinada STOP. El campo destinado a número de identificación debe aparecer en blanco.

H. Problemas propuestos

a) Programar

$$Y = \frac{(A + B)^2}{(A - B)} \qquad \text{si } A - B > 0$$

$$Y = \frac{A + 2(A+B)}{A - B} \qquad \text{si } A - B < 0$$

$$Y = 5(A + B) - B \qquad \text{si } A - B = 0$$

b) Programar

$$F = \frac{7(A + B) + C}{C + A + D} \qquad \text{si } C + B < A$$

$$F = \frac{C - A}{(D - B)\,A + C} \qquad \text{si } C + B \geq A$$

c) Programar

$$F = A(A + B - |C + D|)\ 1/2$$

d) Programar

$$Z = |X| - 2|X - |Y||$$

e) Se pide el valor de L al terminar el siguiente programa:

```
      L = 3
      L = (L / 4) * 4 + L
      IF (L - 15) 1,6,6
    1 L = L + 2
      IF (L - (7 * L) / 4) 6,4,4
    4 L = L - 1
    6 STØP
      END
```

f) Programar

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \pm \cdots$$

detener el cálculo cuando el último término sea, en valor absoluto, menor que $10^{-5}$.

g) Programar

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

igual criterio de detención que en el problema anterior.

h) Programar el cálculo de $\pi$ a base de:

i) $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \pm \cdots$

ii) $\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots$

iii) $\frac{\pi^2}{12} = 1 - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} \pm \cdots$

iv) $\frac{\pi^2}{8} = 1 + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \cdots$

Detener el proceso cuando el valor absoluto del último término calculado sea menor que $10^{-7}$.

i) Programar

$$\log 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} \pm \cdots$$

igual criterio de detención que en el problema h).

j) Programar el cálculo de e a base de:

i) $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots$

ii) $\frac{1}{e} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \pm \cdots$

detener el cálculo cuando el valor absoluto del último término sea menor que $10^{-8}$.

k) Programar

$$(1 + x)^n = 1 + nx + \frac{n(n-1)}{2!} x^2 + \frac{n(n-1)(n-2)}{3!} x^3 + \cdots$$

n debe ser entero $> 0$; $x^2 < 1$

l) Programar

$$(1 + x)^{-n} = 1 - nx + \frac{n(n+1)}{2!} x^2 - \frac{n(n+1)(n+2)}{3!} x^3 \pm \cdots$$

n debe ser entero > 0 $\quad x^2 < 1$

m) Programar

$$(1 + x)^{\frac{1}{2}} = 1 + \frac{1}{2}x - \frac{1\cdot 1}{2\cdot 4} x^2 + \frac{1\cdot 1\cdot 3}{2\cdot 4\cdot 6} x^3 - \frac{1\cdot 1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 8} x^4 \pm \cdots$$

debe cumplirse que $x^2 \leq 1$, detener el proceso cuando el valor absoluto del último término sea menor o igual que $10^{-8}$.

n) Programar

$$\log x = \frac{x-1}{x} + \frac{1}{2} \frac{(x-1)^2}{x} + \frac{1}{3} \frac{(x-1)^3}{x} + \cdots$$

igual criterio de detención que en el problema anterior.

o) $$\log \left(\frac{x-1}{x-1}\right) = 2 \left[\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \frac{1}{7x^7} + \cdots\right]$$

debe cumplirse que $x^2 > 1$. Igual criterio de detención que en el problema m).

## 3. Proposiciones de entrada/salida (input/output)

Permiten al programador transferir información desde dispositivos de entrada a memoria y de ésta hacia dispositivos de salida.

Los elementos que participan en el proceso de entrada/salida son:

a) Conjunto de datos (DATA SET)

b) Dispositivo de entrada/salida

c) Proposición de entrada o salida que contiene además la lista de variables (lista de entrada/salida)

d) Proposición de formato que indica cómo están agrupados los datos o cómo deben agruparse.

Para referirse al conjunto de datos, es decir, para indicar al computador cuál es y donde está el conjunto que se desea transferir, se utiliza una constante entera sin signo o variable entera, sin signo, sin subíndices. Aún cuando de esta forma se identifica simbólicamente la unidad, se ha preferido llamar a esa constante o variable "número de referencia del conjunto de datos" dado que es a éste al que se desea indicar y no a un dispositivo en especial.

Es necesario tener en cuenta que estos números de referencia se asignan en forma permanente a los dispositivos con que cuenta la instalación, esto significa que las asignaciones hechas pueden variar de un computador a otro de acuerdo con la configuración que tenga cada uno.

En los ejemplos que se han visto, como asimismo en los que se desarrollarán más adelante, se utilizan las siguientes asignaciones:

| Número de Referencia | Tipo de dispositivo permitido | Tipo de operación permitida |
|---|---|---|
| 1 | Lectora de tarjetas | Entrada |
| 3 | Impresora | Salida |

Hay dos tipos de proposiciones de entrada/salida: las proposiciones secuenciales y las de acceso directo (no disponibles las últimas en el Soporte Básico de Programación FORTRAN IV Básico). Las proposiciones secuenciales proporcionan las facilidades para la ubicación, selección y recuperación de datos organizados secuencialmente. Son independientes del dispositivo, pues un conjunto de datos secuenciales puede residir en cualquier tipo de volumen.

Las proposiciones de acceso directo proporcionan facilidades para la ubicación, selección y recuperación de datos en un orden especificado por el usuario. Son sólo válidas cuando el conjunto de datos va a residir o ya reside en dispositivos de memoria de acceso directo.

A. Lista de entrada/salida

Cuando se ejecuta una proposición de entrada, la información es llevada a memoria a ubicaciones que no son necesariamente contiguas. Si se trata de recuperar información desde memoria, casi siempre ella es reunida desde distintas ubicaciones y emitida al exterior mediante una proposición de salida. Esas posiciones de memoria donde quedará la información o desde donde se recuperará, se especifican con un conjunto de nombres simbólicos que constituyen la lista de entrada/salida.

Los nombres simbólicos que puede contener la lista son: nombres de variables, con o sin subíndice, y nombres de arreglos. En el primer caso se transferirá un solo valor por cada nombre, en cambio si se trata de nombres de arreglos se transferirán tantos datos como elementos tenga cada arreglo. La cantidad de elementos de un arreglo se determina a base de la especificación hecha en la proposición DIMENSION y el orden en que se transfieren es el que corresponde al almacenamiento en memoria.

La transferencia de arreglos en la forma vista tiene dos limitaciones: deben entrar o salir de memoria todos los elementos del arreglo y además deben hacerlo en el orden en que quedan en memoria. Si se desea eliminar esas restricciones es necesario hacer uso de la estructura DO (DO implícito) para listas de entrada/salida. Esta estructura va separada por coma a continuación de una lista (o sublista) de entrada/salida, ambas encerradas entre paréntesis.

Ejemplo 31:

a) Lista de datos. Se desea imprimir: las variables x,y,z el arreglo lineal C con 10 elementos, el primer y el quinto elementos del arreglo lineal D.

```
X,Y,Z,C,D(1),D(5)
```

b) Lista de datos con DO implícito. El mismo caso anterior.

```
X,Y,Z,(C(I),I=1,10),D(1),D(5)
      sublista
```

c) Variables simples con DO implícito. Se desea repetir la transferencia de X,Y,Z por cada elemento de C.

```
(X,Y,Z,C(I),I=1,10)
 sublista
```

d) Se desea transferir cada elemento impar del arreglo junto con el valor del índice correspondiente

```
(I,C(I),I=1,10,2)
sublista
```

e) Se desea transferir un número N de elementos siendo N menor o igual que el número máximo de elementos.

```
N,(C(I),I=1,N)
```

f) Transferencia de un arreglo bidimensional

i) ((E(I,J),I=1,5),J=1,6)

varía el índice I más rápido que el índice J, esto es, corresponde al orden que tienen los elementos en memoria.

ii) ((E(I,J),J=1,6),I=1,5)

varía el índice J más rápido que el índice I

g) Transferencia de un arreglo tridimensional

(((F(I,J,K),I=1,3),J=1,2),K=1,4)

varía rápido I, más lento J y más lento aún K.

B. Proposición READ

Esta proposición permite introducir datos a la memoria principal.

i) Estructura de la proposición.

READ (a,b,ERR=c,END=d) lista

donde:

a: es un número de referencia de conjunto de datos (data set), dado en forma de constante entera sin signo o variable entera sin signo, sin subíndices.

b: es opcional y si se utiliza puede ser el número de identificación de la proposición FORMAT, que describe el registro que se va a leer, el nombre de un arreglo que contiene las especificaciones de formato o un nombre definido en una proposición NAMELIST. Si b no se utiliza significa que los datos serán leidos sin formato.

ERR=c: es opcional y si se utiliza, c es el número de identificación de una proposición contenida en el mismo programa que contiene a READ. Esa proposición será la meta de salto en caso de detectar error durante la transferencia.

END=d: es opcional y si se utiliza, d es el número de identificación de una proposición contenida en el mismo programa que contiene a READ. Esa proposición será la meta de salto en caso de detectar el término del conjunto de datos.

lista: es opcional y corresponde a una lista de entrada.

ii) Función. Se transfieren datos a la memoria principal desde el conjunto de datos identificado por a. Si se lee sin formato, los datos leídos deben haber sido grabados anteriormente con una proposición WRITE sin formato. En caso de detectar error o término del conjunto de datos (fin de archivo) la próxima proposición a ejecutar está señalada por c o d respectivamente, siempre que se utilicen estos parámetros. Si la lista no es colocada, se efectúa el salto de un registro. Los parámetros ERR=c y END=d se pueden colocar en cualquier orden cuando se usan ambos.

Ejemplo 35:

```
1)  READ(1,51,ERR=20,END=50)A,B,C,D
2)  READ(N,57,END=100)I,J,(A(I),I=1,50)
3)  READ(1,60)((B(I,J),I=1,10),J=1,15)
4)  READ(10)N,(C(I),I=1,N)
5)  READ(11,65)(D(I),E(I),I=1,50),(F(J),J=1,30)
```

C. Proposición WRITE

Esta proposición permite sacar datos desde la memoria principal.

i) Estructura de la proposición.

```
WRITE (a,b) lista
```

donde:

a: es un número de referencia de conjunto de datos (ver proposición READ)

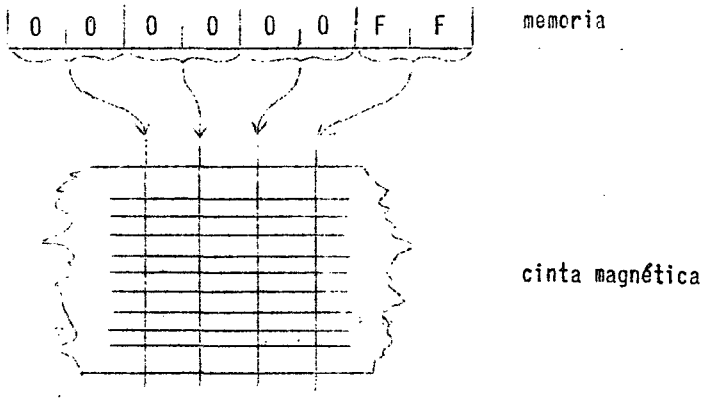
b: es opcional y si se utiliza puede ser: el número de identificación de la proposición FORMAT que describe el registro que se va a imprimir (grabar), el nombre de un arreglo que contiene las especificaciones de formato o un nombre definido en una proposición NAMELIST. Si b no se utiliza, significa que los datos serán grabados sin formato.

lista: es opcional y corresponde a una lista de salida

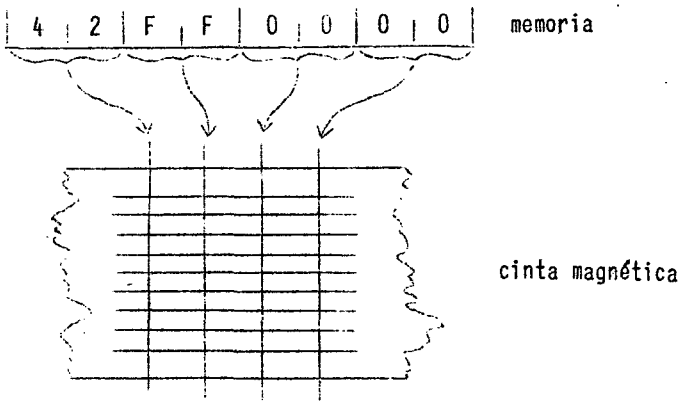
ii) Función. Se transfieren datos desde la memoria principal al conjunto de datos identificado por a. Si se graba sin formato, los datos grabados ocupan cuatro u ocho bytes según sea entero, real de precisión simple o real de precisión doble.

Ejemplo 36:

1) Grabación del dato entero 255 en cinta magnética sin usar formato.




2) Grabación del dato real 255, en cinta magnética, sin usar formato;




Si se hubiera grabado con formato cada carácter ocuparía un byte en la cinta magnética, de aquí que, números que tengan más de cuatro caracteres ocuparán más bytes en cinta magnética o en otro dispositivo magnético que si se graba sin formato.

Si no se pone lista de salida significa que se hará uso de la proposición FORMAT en la cual se tiene como argumento a constantes literales (ver "Constante literal en una proposición FORMAT").

Ejemplo 37:

```
 i)  WRITE (a,51)A,B,C,D
ii)  WRITE (3,52)A,B,C,(D(I),I=1,50)
iii) WRITE (11)A,B,C,(D(J),E(J),J,J=1,100)
iv)  WRITE (11,53)N,(A(K),K=1,N)
 v)  WRITE (3,54)
```

Ejemplos de READ y WRITE haciendo uso de arreglos de códigos de formato y de la proposición NAMELIST, se pueden ver en el capítulo "Otras formas de READ Y WRITE".

D. Proposición FORMAT

Aún cuando esta proposición no es de entrada/salida, sino que de especificación, es conveniente analizarla en este capítulo, pues ella está íntimamente ligada a las proposiciones de entrada/salida READ y WRITE.

i) Estructura de la proposición. Tiene tres formas básicas

X FORMAT $(C_1,C_2, \ldots ,C_n)$

X FORMAT $(C_{11},C_{12}, \ldots ,C_{1n}/C_{21},C_{22}, \ldots ,C_{2n}/ \ldots /C_{n1},C_{n2}, \ldots ,C_{nn})$

X FORMAT $(C_{11},C_{12}, \ldots ,C_{1n},(C_{21},C_{22}, \ldots ,C_{2n}),\ldots,(C_{n1},C_{n2},\ldots,C_{nn})\ldots)$

donde:

X: es el número de identificación de la proposición FORMAT

$C_i$ y $C_{ij}$: son códigos de formatos, esto es, son aquellos elementos mediante los cuales se describen campos de datos o especifican características que permiten la transferencia de información.

ii) Función. La proposición FORMAT permite describir los datos que van a ser transferidos con proposiciones READ y/o WRITE. De acuerdo a la estructura de FORMAT utilizada queda definido el (los) registro (s) FORTRAN, esto es, se especifica la cantidad de datos que tendrá cada registro físico.

1) Primera forma básica

X FORMAT $(C_1, C_2, \ldots, C_n)$

| REGISTRO FORTRAN |

Todos los datos descritos con los códigos de formato que están entre paréntesis (argumento del FORMAT) deben estar contenidos en un REGISTRO FORTRAN, por ejemplo, una tarjeta o una línea de impresión.

Si hay más códigos de formato que datos, los códigos sobrantes se ignoran. Si hay menos códigos de formato que datos se define un nuevo registro FORTRAN y la descripción de los campos se inicia otra vez con el código del extremo izquierdo (paréntesis izquierdo).

2) Segunda forma básica

X FORMAT$(C_{11}, \ldots, C_{1n}/C_{21}, \ldots, C_{2n}/\ldots/C_{n1}, \ldots, C_{nn})$

| REGISTRO FORTRAN | REGISTRO FORTRAN | REGISTRO FORTRAN |

Todos los datos descritos con los códigos de formato que están entre el paréntesis izquierdo y el primer operador de división (slash), deben estar contenidos en un registro FORTRAN, todos los que están descritos con los códigos que figuran entre el primer operador de división y el segundo, en un nuevo registro FORTRAN y así sucesivamente.

Si hay más códigos o menos códigos se aplican las normas indicadas en la "Primera forma básica".

Se pueden saltar registros de entrada o introducir registros de salida compuestos por blancos usando operadores de división consecutivos. Si hay n operadores consecutivos al comienzo o al final del argumento del FORMAT, se saltan o insertan n registros. Si aparecen n operadores consecutivos en cualquiera otra parte del argumento, se saltan o insertan n-1 registros.

Ejemplo 32:

X FORMAT(///...////)

se saltan o insertan tres registros al iniciarse la transferencia y se saltan o insertan cuatro registros cuando se detectan los cuatro operadores que figuran al término del argumento. Si la transferencia continúa, el control de formato se inicia nuevamente en el paréntesis izquierdo.

X FORMAT(.../////...)

los cuatro operadores que figuran entre los códigos de formato permiten saltar o insertar tres registros.

La utilidad principal que tienen los operadores de división consecutivos es poder dejar líneas en blanco en la salida, lo que permite darle más claridad a los resultados impresos.

3) Tercera forma básica

X FORMAT($C_{11}$,...,$C_{1n}$,($C_{21}$,...,$C_{2n}$),...,($C_{n1}$,...,$C_{nn}$)...)

|← PRIMER REGISTRO FORTRAN →

| REGISTROS
|←FORTRAN →
| SIGUIENTES |

Todos los datos descritos con los códigos de formato, que están entre el primer paréntesis izquierdo y el último paréntesis derecho (primer nivel), deben estar contenidos en el PRIMER REGISTRO FORTRAN. Si hay más códigos que datos, los códigos de formato sobrantes se ignoran. Si hay menos códigos, la descripción de los datos se obtiene con los códigos que están en el último grupo encerrado en paréntesis de segundo nivel, más los códigos que figuran hasta el último paréntesis derecho. Este conjunto de códigos define los REGISTROS FORTRAN SIGUIENTES.

Dentro de los paréntesis de segundo nivel se acepta un tercer nivel de paréntesis. El cuarto nivel no es aceptado.

Además de permitir la definición de registros FORTRAN, los paréntesis se usan también para encerrar un grupo de códigos de formatos que se repiten un número a de veces. El factor de repetición se especifica inmediatamente antes del paréntesis izquierdo que encierra el grupo.

Ejemplo 33:

X FORMAT($C_1$,2($C_2$,$C_3$),...,$C_n$) es idéntico a
X FORMAT($C_1$,$C_2$,$C_3$,$C_2$,$C_3$,...,$C_n$)
X FORMAT(...,a(...,a(...),...,a(...),...),...) es válido
X FORMAT(...,a(...,a(...,a(...),...),...),...) no es válido

4) Carácter de control de carro

El primer carácter de un registro de salida, cuando se trata de impresión, es considerado carácter de control del carro de la impresora, esto es, no se imprime y se interpreta de acuerdo con la siguiente tabla:

| Carácter | Significado (acción del carro de la impresora) |
|---|---|
| blanco | avanza una línea antes de imprimir |
| 0 | avanza dos líneas antes de imprimir |
| 1 | avanza a la primera línea de la página siguiente |
| + | no avanza |

Cualquier otro carácter se interpreta como blanco (Ø). La forma más común de especificar el carácter es colocándolo entre apóstrofos al comienzo del argumento del FORMAT.

Ejemplo 34:

X FORMAT('0',...)

con lo cual se indica que el carro de la impresora debe avanzar dos líneas antes de que se realice la impresión.

E. Códigos de formato

Los códigos de formato son:

I para describir campos de datos enteros
F para describir campos de datos reales
E para describir campos de datos reales con exponente E
D para describir campos de datos reales con exponente D
L para describir campos de datos lógicos
G para describir campos de datos enteros, reales o lógicos
A para describir campos de caracteres

Z para describir campos de datos hexadecimales

H para indicar datos literales

'literal' para indicar datos literales

X para indicar que un campo va a ser saltado en entrada o llenado con blancos en salida

T para especificar la posición en un registro FORTRAN a partir de la cual se va a realizar la transferencia de datos

En las estructuras de códigos de formato se utilizan los símbolos siguientes:

a: es una constante entera sin signo, opcional, usada para indicar el número de veces que se ocupará el mismo código. Si se omite, el código se ocupa una sola vez.

w: es una constante entera sin signo, distinta de cero, que especifica la cantidad de caracteres que tiene el campo, esto es, si hay signo, punto, etc., deben considerarse en w.

d: es una constante entera sin signo, que especifica el número de lugares decimales a la derecha del punto decimal, esto es, indica la cantidad de dígitos de la parte fraccionaria de un número.

p: es opcional y representa un factor de escala de la forma nP, en que n es una constante entera, con o sin signo. Para usarlo se aplica la fórmula siguiente:

$$\text{CANTIDAD EXTERNA} = \text{CANTIDAD INTERNA} * 10^{n}$$

s: es una constante entera sin signo, que especifica el número de dígitos significativos en salida o la parte fraccionaria en entrada.

r: es una constante entera sin signo, que especifica una posición dentro de un registro FORTRAN.

1. Código de Formato I

i) Estructura del código

a I w

donde:

a: es factor de repetición

w: es la cantidad de caracteres

ii) Función: Se utiliza para transferir datos enteros. Si es ENTRADA de datos, los blancos que encabezan la información, los que estén intercalados con los dígitos o en el extremo derecho son considerados como ceros. La magnitud no debe exceder la capacidad máxima definida para constante entera. Si es salida de datos se presentan dos posibilidades, además de la normal, que es cuando w es igual a la cantidad de caracteres que tiene el dato. Una posibilidad es que w sea mayor que la cantidad de caracteres, en cuyo caso las posiciones excedentes de la izquierda se rellenan con blancos. La otra posibilidad es que w sea menor que la cantidad de caracteres, situación en la cual se imprimen w asteriscos.

Ejemplo 38:

Leer seis valores enteros perforados en una tarjeta. Los dos primeros tienen tres y cinco caracteres respectivamente. Los cuatro restantes tienen cuatro caracteres cada uno. Imprimir los seis valores de tal manera que queden tres datos por línea. Los tres primeros con seis caracteres cada uno y los tres siguientes con siete caracteres cada uno.

```
C EJEMPLO 38.
C USØ DE CØDIGØ DE FØRMATØ I
      READ (1,51) I,J,K,L,M,N
      WRITE(3,52) I,J,K,L,M,N
      STØP
   51 FØRMAT (I3,I5,4I4)
   52 FØRMAT ('Ø',3I6/' ',3I7)
      END
```

El espacio en blanco (Ø) que aparece entre apóstrofos se utiliza como carácter de control de carro e indica que avance una línea antes de imprimir.

Suponiendo que los datos de entrada sean:

Columna 1 de la tarjeta
↓
Ø10-32ØØØ4Ø523ØØØ-27Ø+28

Observación:

El carácter Ø se ha utilizado para representar el espacio en blanco tal como en el formato 52 del ejemplo 38. El carácter Ø significa que no se perfora nada en entrada y en salida no se imprime nada en esas posiciones.

El resultado de la impresión será:

posición 1 de la línea

```
ØØØØ10Ø-3200ØØØ405
ØØØ2300ØØØØ-27ØØØØØ28
```

Ejemplo 39:

Los mismos datos del ejemplo 38 están perforados ahora, los dos primeros con cinco caracteres cada uno en la primera tarjeta, los dos siguientes con tres y cuatro caracteres respectivamente en la segunda tarjeta y los dos últimos con cinco caracteres cada uno en la tercera tarjeta. Al imprimirlos se desea que el primer dato quede en la primera línea y dos datos en cada una de las líneas siguientes.

```
C EJEMPLØ 39.
C USØ DE CØDIGØ DE FØRMATØ I
      READ (1,51) I,J,K,L,M,N
      WRITE(3,52) I,J,K,L,M,N
      STØP
   51 FØRMAT (2I5/I3,I4)
   52 FØRMAT ('Ø',I3/'Ø',I4,I6/('Ø',I2,I5))
      END
```

La perforación de los datos será la siguiente:

columna 1

```
ØØØ10-32          1a tarjeta
4Ø523             2a tarjeta
ØØ-27ØØØØ28       3a tarjeta
```

y la impresión será:

posición 1

```
Ø10               1a línea
****ØØØ405         2a línea
**ØØ-27           3a línea
28                4a línea
```

La impresión se realiza de acuerdo con la correspondencia siguiente:

| Variable | Código de formato |
|---|---|
| I | I3 |
| J | I4 |
| K | I6 |
| L | I2 |
| M | I5 |
| N | I2 |

Nótese que la variable J, cuyo valor es -3200, aparece impresa con ****, esto es, no se contemplaron los caracteres suficientes en w y se imprimieron w asteriscos. Lo mismo ocurre con la variable L, que tiene valor 2300 y w indica impresión de dos caracteres.

2. Códigos de formato F, E y D

i) Estructura del código

p a F w.d
p a E w.d
p a D w.d

donde:

p: es factor de escala
a: es factor de repetición
w: es cantidad de caracteres
d: es parte fraccionaria

ii) Función. Se utilizan para transferir datos reales. Estos no deben exceder la magnitud máxima que corresponde a contantes reales.

En ENTRADA, el dato puede tener, opcionalmente, exponente que debe estar precedido por una constante de al menos un dígito, con o sin signo, con o sin punto decimal. Si el dato tiene punto decimal, éste tiene prioridad sobre la indicación dada por d en el código de formato. Si el dato tiene exponente decimal E, D o constante entera con signo y en el código de formato se coloca factor de escala, éste no tiene efecto sobre el dato. Los espacios en blanco que figuren en el dato se consideran ceros.

Los códigos F, E y D se pueden usar para leer indistintamente datos con exponente E, D o constante entera con signo. En cualquier caso, tiene prioridad el tipo de la variable leída.

En SALIDA, en el código de formato F, en el valor w, están incluidos los dígitos de la parte entera, los de la parte fraccionaria, el punto decimal y el signo, si el valor es negativo. Los códigos E y D se pueden intercambiar para imprimir reales de precisión simple o doble, el resultado es impreso con la letra que corresponde al tipo de la variable que figura en la lista de salida. Deben contemplarse en w, a menos que se utilice factor de escala P, posiciones para: signo (si el dato es negativo), punto decimal, dígitos significativos y la letra D o E seguida de constante entera de dos dígitos y signo. Si hay espacio suficiente contemplado en w, aparece impreso el dígito cero antes del punto decimal. Como norma general, la constante 7 se debe sumar a la cantidad de dígitos significativos que se desea imprimir.

Si se utiliza factor de escala P con código de formato E o D, no tiene efecto sobre la magnitud del dato impreso sino en su estructura dado que el punto decimal se desplaza a la izquierda o a la derecha y el exponente disminuye en el primer caso y aumenta en el segundo.

Ejemplo 40:

Se tienen los siguientes datos:

12345-78987654321

El programa que figura a continuación (REAL*8 C,D declara las variables C y D de doble precisión):

```
C EJEMPLØ 40.
C USØ DE CØDIGØS DE FØRMATØ E Y D
C CØN FACTØR DE ESCALA P
      REAL*8 C,D
      READ (1,51) A,B,C,D
      WRITE(3,52) A,B,C,D
      WRITE(3,53) A,B,C,D
      STØP
   51 FØRMAT(2E5.3,2D5.3)
   52 FØRMAT(' ',E10.5,E14.5)
   53 FØRMAT(' ',1P2E13.4,-1P2D13.4)
      END
```

imprime de acuerdo con los datos leidos, los resultados que se indican:

```
.12345EØ2ØØ-0.78980EØ1
.76543DØ2ØØØ0.21000DØ2
ØØØ1.2345EØ1ØØ-7.8980EØ00ØØØ0.0765DØ03ØØØ0.0210DØ03
```

Ejemplo 41:

Se tienen los siguientes datos:

```
Ø45.3ØØ28Ø14E+1Ø4.D2Ø2.+1ØØ3-2
Ø453ØØ453ØØ453ØØ453Ø
```

El programa que figura a continuación:

```
C EJEMPLØ 41.
C USØ DE CØDIGØ DE FØRMATØ F
      READ (1,51)A,B,C,D,E,F
      WRITE(3,52)A,B,C,D,E,F
      READ (1,53)A,B,C,D
      WRITE(3,54)A,B,C,D
      STØP
   51 FØRMAT(2F5.2,4F5.0)
   52 FØRMAT(6F8.2)
   53 FØRMAT(1PF5.2,2PF5.2,-1PF5.2,-2PF5.2)
   54 FØRMAT(4F10.4)
      END
```

imprime de acuerdo con los datos leidos, los resultados que se indican.

```
ØØ45.30ØØØØ2.80ØØ140.00ØØ400.00ØØØ20.00ØØØØ0.03
ØØØ4.5300ØØØØ0.4530ØØ453.0000Ø4530.0000
```

Ejemplo 42:

Se tienen los siguientes datos:

```
15.995.4-152.5D-1          1a tarjeta
15.995.4-152.5E-1          2a tarjeta
```

El programa que figura a continuación lee estos datos; la primera tarjeta, que contiene un dato con exponente D, la lee con código de formato E y la segunda, que contiene un dato con exponente E, la lee con código de formato D. Se hace uso nuevamente de la proposición REAL*8 para definir las variables X, Y y Z, de doble precisión.

```
C EJEMPLØ 42.
C USØ DE CØDIGØS DE FØRMATØ E Y D
      REAL*8 X,Y,Z
      READ (1,51)A,B,C
      WRITE(3,52)A,B,C
      READ (1,53)X,Y,Z
      WRITE(3,54)X,Y,Z
      STØP
   51 FØRMAT(2E5.2,E7.1)
   52 FØRMAT(F8.1,2E12.3)
   53 FØRMAT(2D5.2,D7.1)
   54 FØRMAT(3D12.3)
      END
```

Los resultados que se imprimen son los que siguen:

```
ØØØ16.0ØØØ0.540EØ00ØØØ0.525EØ01
ØØ0.160DØ02ØØØ0.540DØ00ØØØ0.525DØ01
```

Ejemplo 43:

Un profesor tiene un curso de 20 alumnos. Cada uno de ellos tiene tres notas correspondientes a pruebas parciales. El profesor ha preparado una tarjeta por alumno con la siguiente estructura:

| | | |
|---|---|---|
| Número del alumno | (NA) | columnas 1 y 2 |
| Nota 1 | (C1) | columnas 3 y 4 |
| Nota 2 | (C2) | columnas 5 y 6 |
| Nota 3 | (C3) | columnas 7 y 8 |

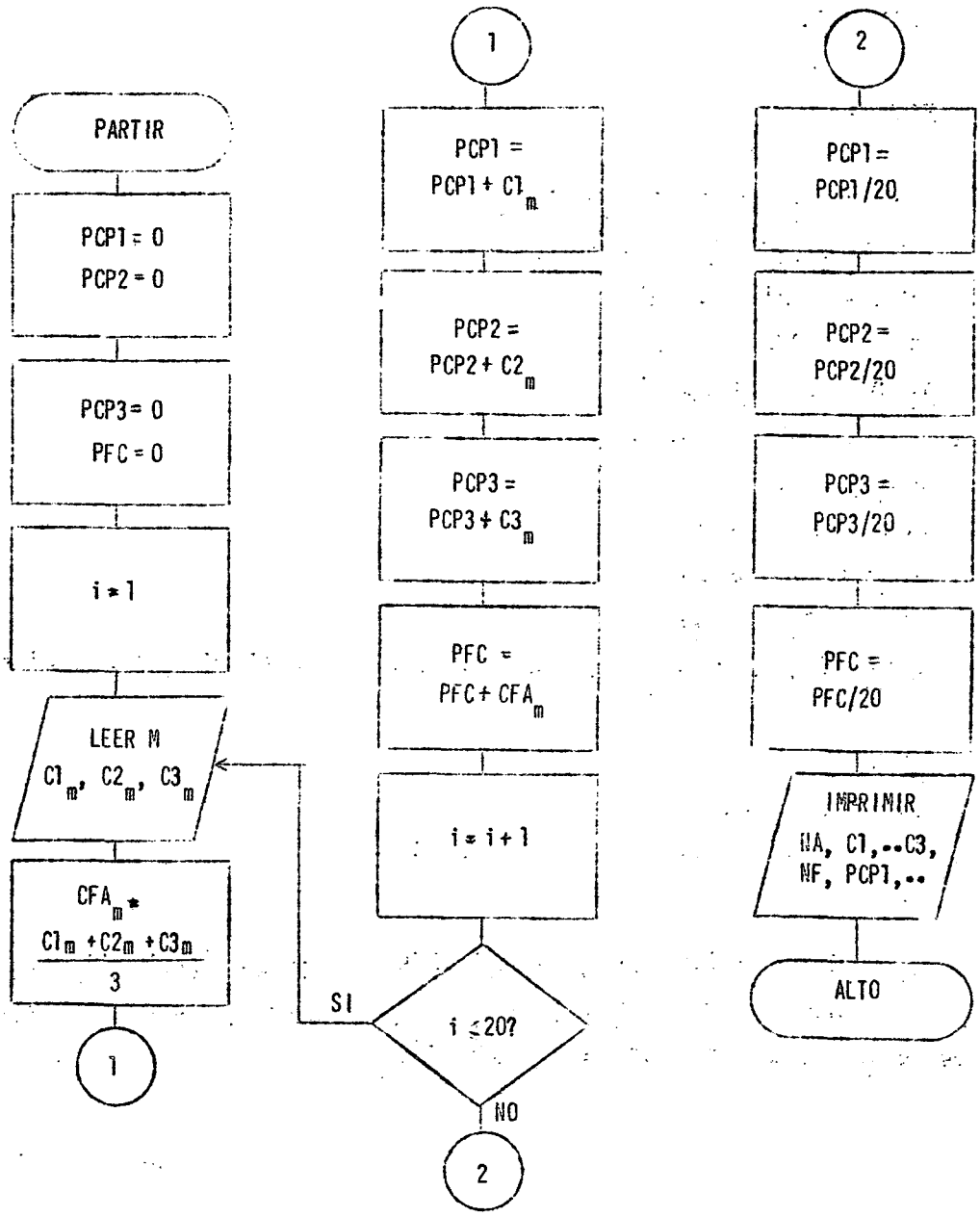
Se desea programar el cálculo de:

- promedio por alumno (nota final) CFA
- promedio del curso en cada prueba PCP
- promedio final del curso PFC

Se supondrá que el número del alumno varía de 1 a 20, pero que las tarjetas pueden entrar desordenadas.

En una primera solución se considera que están todas las tarjetas y en una segunda solución que faltan tarjetas; en este último caso se terminará la lectura cuando se detecte una tarjeta en blanco.

a) Primera solución. El número del alumno servirá como índice para cuatro arreglos, que tendrán las tres notas parciales y la nota final. En un solo ciclo se incluye la lectura de los datos de un alumno, el cálculo de su nota final y la acumulación para el cálculo del promedio del curso.

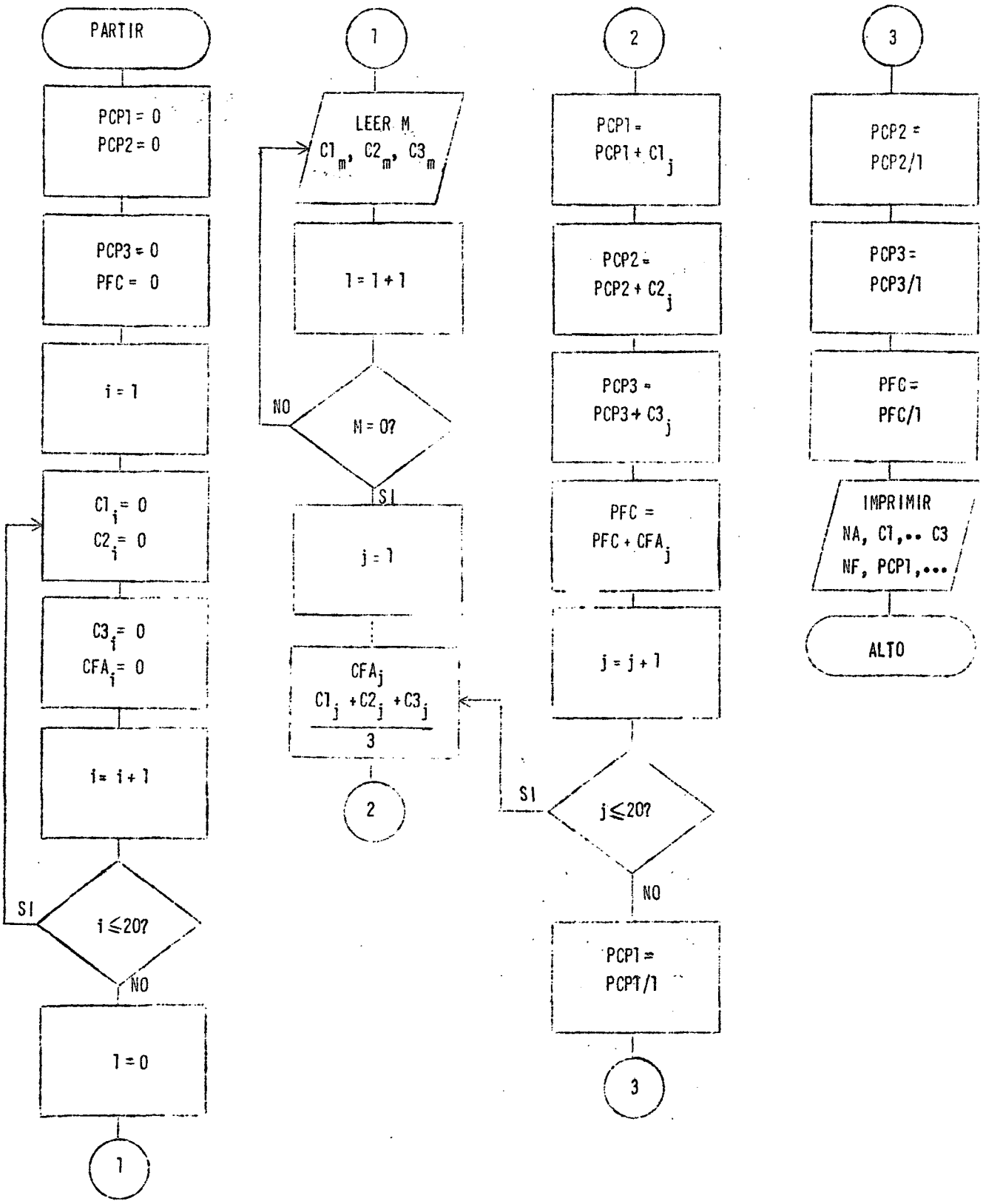


Observación:

A pesar de que el programa que se obtiene se ha minimizado en cuanto a número de instrucciones, el hecho de tener una proposición de lectura que es lenta, dentro de un ciclo en que aparecen proposiciones de asignación, hace que todo el proceso sea lento. Es preferible tener un ciclo de lectura en que se almacene toda la información para efectuar a continuación todos los cálculos.

```
C EJEMPLØ 43. (SØLUCIØN A)
C CALCULØ DE NØTAS DE UN CURSØ
      DIMENSIØN C1(20),C2(20),C3(20),CFA(20)
      PCP1 = 0.
      PCP2 = 0.
      PCP3 = 0.
      PFC  = 0.
      DØ 10 I=1,20
      READ (1,51) N,C1(N),C2(N),C3(N)
      CFA(N) = (C1(N)+C2(N)+C3(N))/3.
      PCP1 = PCP1 + C1(N)
      PCP2 = PCP2 + C2(N)
      PCP3 = PCP3 + C3(N)
   10 PFC  = PFC  + CFA(N)
      PCP1 = PCP1 / 20.
      PCP2 = PCP2 / 20.
      PCP3 = PCP3 / 20.
      PFC  = PFC  / 20.
      WRITE (3,52) (I,C1(I),C2(I),C3(I),CFA(I),I=1,20),PCP1,PCP2,PCP3,PF
     2C
      STØP
   51 FØRMAT(I2,3F2.1)
   52 FØRMAT(20(' ',I4,4F7.2/)//F12.2,3F7.2)
      END
```

b) Segunda solución. Dado que se supone que las tarjetas pueden entrar desordenadas, nuevamente se utilizará el número del alumno como índice de los arreglos en que se almacenará la información.


PARTIR
PCP1 = 0
PCP2 = 0
PCP3 = 0
PFC = 0
i = 1
C1i = 0
C2i = 0
C3i = 0
CFAi = 0
i = i + 1
i ≤ 20?
SI
NO
l = 0
1
1
LEER M
C1m, C2m, C3m
l = l + 1
M = 0?
NO
SI
j = 1
CFAj
(C1j + C2j + C3j) / 3
2
2
PCP1 =
PCP1 + C1j
PCP2 =
PCP2 + C2j
PCP3 =
PCP3 + C3j
PFC =
PFC + CFAj
j = j + 1
j ≤ 20?
SI
NO
PCP1 =
PCP1/l
3
3
PCP2 =
PCP2/l
PCP3 =
PCP3/l
PFC =
PFC/l
IMPRIMIR
NA, C1,.. C3
NF, PCP1,...
ALTO

```
C EJEMPLØ 43.(SØLUCIØN B)
C CALCULØ DE NØTAS DE UN CURSØ
      DIMENSIØN C1(20),C2(20),C3(20),CFA(20)
      PCP1 = 0.
      PCP2 = 0.
      PCP3 = 0.
      PFC  = 0.
      DØ 10 I=1,20
      C1(I) = 0.
      C2(I) = 0.
      C3(I) = 0.
   10 CFA(I)= 0.
      L = 0
   11 READ (1,51) N,C1(N),C2(N),C3(N)
      L = L + 1
      IF (N.NE.0) GØ TØ 11
      DØ 12 J=1,20
      CFA(J) =(C1(J)+C2(J)+C3(J))/3.
      PCP1 = PCP1 + C1(J)
      PCP2 = PCP2 + C2(J)
      PCP3 = PCP3 + C3(J)
   12 PFC  = PFC  + CFA(J)
      PCP1 = PCP1 / L
      PCP2 = PCP2 / L
      PCP3 = PCP3 / L
      PFC  = PFC  / L
      WRITE(3,52)(I,C1(I),C2(I),C3(I),CFA(I),I=1,20)
      WRITE(3,53)PCP1,PCP2,PCP3,PFC
      STØP
   51 FØRMAT(I2,3F2.1)
   52 FØRMAT(' ' I4,4F7.2)
   53 FØRMAT(///F12.2,3F7.2)
      END
```

Observación: La variable L se utiliza para contabilizar la cantidad de tarjetas leidas y calcular, con ese valor, los promedios que se desean.

3. Código de formato L

i) Estructura del código

a L w

donde:

a: es factor de repetición

w: es cantidad de caracteres

ii) Función. Se utiliza para transferir datos lógicos. En ENTRADA el dato debe estar formado por: un blanco al menos, seguido por la letra T o la letra F y a continuación cualquier tipo de caracteres. La letra T causa que sea asignado el valor VERDADERO (TRUE), a la variable que figura en la lista de entrada/salida. La letra F causa que sea asignado el valor FALSO (FALSE) a dicha variable. En SALIDA se imprime la letra T o la F, según sea el valor de la variable, verdadero o falso respectivamente. La letra se ajusta a la derecha en el campo de salida, precedida por w-1 blancos.

Ejemplo 44:

Se tienen los siguientes valores, perforados en tres tarjetas.

| | |
|---|---|
| T FAL TAL FIN | 1a tarjeta |
| TRUE FALSE T F | 2a tarjeta |
| T+* | 3a tarjeta |

El programa que figura a continuación lee esas tarjetas e imprime los valores leidos:

```
C EJEMPLØ 44.
C USØ DE CØDIGØ DE FØRMATØ L
      LØGICAL A,B,C,D,E*1
      READ(1,51)A,B,C,D
      WRITE(3,52)A,B,C,D
      READ(1,53)A,B,C,D
      WRITE(3,52)A,B,C,D
      READ(1,51)E
      WRITE(3,52)E
      STØP
   51 FØRMAT(4L4)
   52 FØRMAT(' ',2L4,L8,L1)
   53 FØRMAT(2L7,2L2)
      END
```

los resultados que se obtienen impresos son:

```
ØØØTØØØFØØØØØØØTF    1a línea
ØØØTØØØFØØØØØØØTF    2a línea
ØØØT
```

4. <u>Código de formato G</u>

Este código se utiliza en reemplazo de los códigos I,F,E,D o L

i) Estructura del código

p a G w. s

donde:

p: es factor de escala

a: es factor de repetición

w: es cantidad de caracteres

s: es parte fraccionaria en entrada y dígitos significativos en salida.

ii) Función. Permite la transferencia de datos enteros, reales o lógicos, que correspondan a variables de su mismo tipo.

En ENTRADA se mantienen las mismas normas dadas para los códigos I,F,E,D y L. Si las variables son enteras o lógicas, la parte <u>s</u> del código de formato se puede omitir; si se especifica, es ignorada.

En SALIDA, igualmente, se mantienen las mismas normas dadas para cada código en particular. Si las variables son enteras o lógicas, la parte <u>s</u> se puede omitir; en caso contrario, se ignora. Para datos reales, la parte <u>s</u> indica el número de dígitos significativos que se desea imprimir, como también si el dato se desea imprimir con o sin exponente decimal. En este último caso, si el número es mayor o igual que 0.1 y menor o igual que 10**s ($0.1 \leq x \leq 10^{s}$), el número es impreso sin exponente decimal. En caso contrario, se imprimirá con el exponente decimal que le corresponda al tipo de variable, esto es, E o D.

Aún cuando el resultado sea impreso sin exponente decimal, el valor que se imprime aparece desplazado hacia la izquierda cuatro lugares que corresponden a la letra E o D seguida de dos dígitos con o sin signo. Si el exponente es positivo no se imprime el signo pero el espacio queda en blanco y debe reservarse. Lo anterior significa que es necesario contemplar en el valor w los cuatro espacios

del exponente, el del punto decimal, el del signo si el dato es negativo y al menos un dígito que preceda al punto decimal. En total, entonces, w será igual a siete más la cantidad de dígitos significativos que se desea imprimir.

Ejemplo 45:

Suponiendo que se tienen tres tarjetas con los siguientes datos:

| | |
|---|---|
| 105224 | la tarjeta |
| 49535.4E+16.2D-185.3+1 4834 | 2a tarjeta |
| TITØ F | 3a tarjeta |

El programa que figura a continuación contiene dos proposiciones de especificación: LOGICAL U,V para indicar que las variables U y V son lógicas, y REAL*8 D que declara que la variable D es de doble precisión (longitud 8 bytes).

```
C EJEMPLØ 45.
C USØ DE CØDIGØ DE FØRMATØ G
      LØGICAL U,V
      REAL*8 D
      READ (1,51) I,J
      WRITE(3,54) I,J
      READ (1,52) A,B,C,D,E
      WRITE(3,55) A,B,C,D,E
      READ (1,53) U,V
      WRITE(3,53) U,V
      STØP
   51 FØRMAT(G4.2,G3)
   52 FØRMAT(G5.2,3G6.1,1PG5.2)
   53 FØRMAT(G5.1,G2)
   54 FØRMAT(' ',G6.2,G6)
   55 FØRMAT(' ',G6.3,3G10.3,0PG7.2)
      END
```

Con este programa se obtienen los resultados siguientes

| | |
|---|---|
| ØØØ105ØØØ224 | 1a línea |
| ******ØØ54.0ØØØØØ0.620ØØØØØØ853.ØØØØ4.8ØØØØ | 2a línea |
| ØØØTbF | 3a línea |

5. Código de formato Z

i) Estructura del código

a Z w

donde:

a: es factor de repetición

w: es cantidad de caracteres

ii) Función. Permite la transferencia de datos hexadecimales.

En ENTRADA, los blancos que figuren en el campo leido se consideran ceros hexadecimales. Cada byte contiene dos dígitos hexadecimales, de tal manera que si se lee un número impar de dígitos el dato se ajusta a la derecha y se rellena con un cero hexadecimal por la izquierda. Lo mismo ocurre cuando el campo es mayor que el necesario para los caracteres leidos, esto es, se rellena con ceros hexadecimales por la izquierda. Si el campo es menor que el necesario para los caracteres que se leen, se recortan los caracteres de orden superior.

En SALIDA, si el número de caracteres del dato es menor que w, el campo se rellena con blancos por la izquierda. Si el número de caracteres es mayor que w, se pierden los caracteres de orden superior.

Ejemplo 46:

Se tienen las siguientes tarjetas de datos.

```
A BFFA 1869ABCDEF45.5 255    1a tarjeta
AABFAFBFEDFFFFFFFO           2a tarjeta
```

El programa que figura a continuación lee la información de esas tarjetas e imprime. Se declara la variable B como de doble precisión con la proposición REAL*8 B

```
C EJEMPLØ 46.
C USØ DE CØDIGØ DE FØRMATØ Z
      REAL*8 B
      READ (1,51) I,A,B,C,J
      WRITE(3,52) I,A,B,C,J
      READ (1,53) J,A,I
      WRITE(3,54) J,A,I
      WRITE(3,55) J,I
      STØP
```

(Continúa)

(Continuación)

```
51 FØRMAT(2Z4,Z10,F5.3,I3)
52 FØRMAT(' ',I10,Z2,Z14,2Z10)
53 FØRMAT(Z3,Z4,Z11)
54 FØRMAT(' ',Z2/' ',Z3/' ',Z8)
55 FØRMAT(2I10)
   END
```

Los resultados que se obtienen son:

```
ØØØØØØ2571A000001869ABCDEFØØ422D8000ØØ000000FF
AB
AFB
FFFFFFF0
ØØØØØØ171ØØØØØØØØ-16
```

En la primera lectura la variable I se define con los dígitos hexadecimales AOB ajustados a la derecha. Estos corresponden a:

$$A*16^2+0*16^1+B*16^0$$

$$10*256+0+11$$

$$2571$$

que es el valor que se imprime con la primera proposición WRITE.

La operación inversa se efectúa con las variables C y J, que se definen con los valores 45.5 y 255 respectivamente.

$$(45.5)_{10} = (2D.8)_{16} = 0.2D8*16^2$$

la característica será entonces

$$(64+2)_{10} = (66)_{10} = (42)_{16}$$

como el signo es positivo no afecta a este valor y queda

422D8000, que es el valor impreso.

En cuanto a la variable J

$(255)_{10} = (FF)_{16}$ e internamente 000000FF

6. Código de formato A

i) Estructura del código

a A w

donde:

a: es factor de repetición

w: es cantidad de caracteres

ii) Función. Permite la transferencia de caracteres que quedan almacenados o lo están, en formato de carácter. Dado que cada carácter ocupa un byte, la cantidad de ellos que es transferida depende de la longitud con que haya sido definida la variable, la que puede ser de cualquier tipo.

En ENTRADA, si w es menor que la cantidad de bytes (cb) reservada para la variable leida, se leen w caracteres y se ajustan a la izquierda y se rellenan los bytes restantes con blancos. Si w es mayor que la cantidad de bytes reservada se saltan w-cb caracteres y se leen cb caracteres.

En SALIDA, si w es menor que la cantidad de bytes reservada se imprimen w caracteres del extremo izquierdo de la variable. Si w es mayor que la cantidad de bytes, se imprimen cb caracteres precedidos por w-cb blancos.

Ejemplo 47:

La siguiente tarjeta de datos

A+BCDEFG$-*HIJKL1234MNØPQ  única tarjeta

es leida por el programa que figura a continuación

```
C EJEMPLØ 47.
C USØ DE CØDIGØ DE FØRMATØ A
      LØGICAL U
      REAL*8 A
      READ (1,51) I,A,B,U
      WRITE(3,52) I,A,B,U
      STØP
   51 FØRMAT(A4,2A8,A2)
   52 FØRMAT(' ',A2,A12/' ',A4,A6)
      END
```

el cual imprime los resultados que se indican en seguida

| | |
|---|---|
| A+ØØØØØDEFG$-*H | 1a línea |
| 1234ØØØMN | 2a línea |

Ejemplo 48:

Hacer un programa que entregue el gráfico de la función seno x. Los caracteres que se utilizan son los siguientes:

*I

El programa que figura a continuación entrega el gráfico de la función seno x, la cual se obtiene a base de una serie:

```
C EJEMPLØ 48.
C USØ DE CØDIGØ DE FØRMATØ A
C GRAFICØ DE LA FUNCIØN SEN
      DIMENSIØN A(130)
      READ (1,51) BLANCØ,ASTER,EJE
      DØ 10 I=1,130
   10 A(I)=EJE
      WRITE(3,52)A
      DØ 11 I=1,130
   11 A(I)=BLANCØ
      A(65)=EJE
      X = 0.
      DX= 0.07854
      EPSIL = 1.E-5
   12 TER = X
      N   = 1
      SUM = 0.
   13 SUM = SUM + TER
      TER = -TER*X*X/(N+1)/(N+2)
      IF (TER)14,15,15
   14 DELTA =-TER
      GØ TØ 16
   15 DELTA = TER
```

(Continúa)

```
16 IF (DELTA - EPSIL) 18,17,17
17 N = N + 2
   GØ TØ 13
18 Y = SUM + TER
   K = 65 + 50*Y
   A(K)=ASTER
   WRITE(3,52)A
   IF (K - 65)19,20,19
19 A(K)=BLANCØ
   GØ TØ 21
20 A(K)=EJE
21 X = X + DX
   IF (X.LE.6.2832)GØ TØ 12
   STØP
51 FØRMAT(3A1)
52 FØRMAT(' ',130A1)
   END
```

## 7. Constante literal en una proposición FORMAT y código de formato H

### a) Constante literal.

Son cadenas de caracteres alfanuméricos y especiales, incluido el carácter blanco, que van como argumento de la proposición FORMAT encerrados entre apóstrofos. Si aparece un carácter apóstrofo en la cadena de datos, debe ir seguido inmediatamente por otro apóstrofo. Al almacenar el argumento del FORMAT, se hace un análisis de los caracteres y al encontrarse dos apóstrofos seguidos se guarda sólo uno.

En ENTRADA, los caracteres de la tarjeta reemplazan a los caracteres de la cadena de caracteres uno a uno. Si se usan apóstrofos, se lee un número de caracteres igual al que está entre ellos. Si se usa código de formato H, se leen w caracteres.

En SALIDA se imprimen los caracteres que figuran entre apóstrofos o los w caracteres que siguen al código de formato H.

Ejemplo 49:

Se tienen tres tarjetas con los siguientes datos:

| | |
|---|---|
| RESULTADØ | 1a tarjeta |
| SE PRUEBA APØS''TRØFØ | 2a tarjeta |
| RESULTADØ CØDIGØ H EN SALIDA | 3a tarjeta |

El programa que figura a continuación:

```
C EJEMPLØ 49.
C USØ DE LITERAL Y CØDIGØ DE FØRMATØ H
      READ (1,51)
      WRITE(3,51)
      READ (1,52)
      WRITE(3,52)
      READ (1,53)
      WRITE(3,53)
      WRITE(3,54)
      WRITE(3,55)
      STØP
   51 FØRMAT(' PRUEBA DE LITERAL')
   52 FØRMAT(' TITULØ CØN CØDIGØ H  ')
   53 FØRMAT(19H PRUEBA DE CØDIGØ H)
   54 FØRMAT(' LITERAL NØ NECESITA W')
   55 FØRMAT(20H CØDIGØ H NECESITA W)
      END
```

lee las tres tarjetas mencionada e imprime los siguientes resultados:

```
RESULTADØ
SE PRUEBA APØS''TRØFØ
RESULTADØ CØDIGØ H
LITERAL NØ NECESITA W
CØDIGØ H NECESITA W
```

Se puede observar que con las órdenes de lectura se produjo el reemplazo de los argumentos de los FORMAT 51, 52 y 53 por el contenido de las tarjetas. De la lectura e impresión de la segunda tarjeta se concluye que al aparecer dos apóstrofos, como parte de los caracteres encerrados entre otros dos, sólo se almacena uno de ellos; en cambio, al leer dos apóstrofos se almacenan ambos.

8. Código de Formato X

i) Estructura del código

W X

donde:

w: es cantidad de caracteres

ii) Función: El código de formato X causa que en ENTRADA se salten w caracteres y en SALIDA se inserten w blancos.

Ejemplo 50:

Se tiene la siguiente tarjeta de datos

Ø1234.5678.9Ø365,423101

Las primeras cinco columnas definen a la variable N (número de proceso), a continuación es necesario saltarse siete columnas. Las dos columnas siguientes definen a la variable J, las cuatro que siguen a la variable A y las cinco últimas a la variable B.

Se desea calcular:

X = A + B**J

y la impresión debe tener la estructura siguiente:

posición 20
↓
RESULTADOØDELØPROCESOØ#.....
} 3 líneas en blanco

posición 5
↓
DATOS:
{1 línea en blanco

| posición 7 | posición 27 | posición 47 |
|---|---|---|
| ↓ | ↓ | ↓ |
| AØ=ØØØ | BØ=ØØØ | JØ=ØØØ |
| --- | --- | --- |

posición 5
↓
RESULTADO:
}1 línea en blanco
posición 7
↓
XØ=ØØØ
---

El programa que figura a continuación resuelve el problema planteado. En ENTRADA, con la proposición 51 FORMAT(I5,7X,...) se produce la eliminación de las siete columnas que no interesan. En SALIDA, se hace notar que como argumento de la proposición FORMAT pueden aparecer mezclados todos los códigos de formato y constantes literales.

El resultado pedido se ha colocado inmediatamente después del programa.

```
C EJEMPLØ 50.
C USØ DE CØDIGØ DE FØRMATØ X
      READ (1,51) N,J,A,B
      X = A + B ** J
      WRITE(3,52) N,A,B,J,X
      STØP
   51 FØRMAT(I5,7X,I2,F4.1,F5.3)
   52 FØRMAT(' ',19X,'RESULTADØ DEL PRØCESØ #',I5////' ',4X,'DATØS:'//'
     2',6X,'A =',3X,F6.2,8X,'B =',3X,F6.3,8X,'J =',3X,I4///' ',4X,'RESUL
     3TADØ:'//' ',6X,'X =',3X,F11.3)
      END
ØØØØØØØØØØØØØØØØØØØØRESULTADO DEL PROCESO # 1234


ØØØØDATOS:

ØØØØØØA =ØØØ 65.40ØØØØØØØØØB =ØØØ23.101ØØØØØØØØØJ =ØØØ   3


ØØØØRESULTADØ:

ØØØØØØX =ØØØ  12393.383
```

9. Código de Formato T

i) Estructura del código

Tr

donde:

r: indica posición dentro del registro FORTRAN

ii) Función. Se especifica la posición dentro de un registro FORTRAN a partir de la cual se iniciará la transferencia de datos.

La entrada y la salida pueden empezar en cualquier posición usando el código de formato T. Cuando la salida es impresa, la correspondencia entre r y las posiciones de la línea no es exacta debido a que el primer carácter es considerado como control de carro y no se imprime, luego la posición en la línea corresponde a r-1.

Ejemplo 51:

Se tienen los siguientes datos

1234567890987654321

El programa que figura a continuación:

```
C EJEMPLØ 51.
C USØ DE CØDIGØ DE FØRMATØ T
      READ (1,51) XY,MN,JK,AB
      WRITE(3,52) JK,AB,MN,XY
      STØP
   51 FØRMAT(T12,F5.2,T1,I5,T17,I3,T6,E6.3)
   52 FØRMAT(T42,I3,T2,E12.6,T31,I5/' ',T2,E12.5)
      END
```

lee los datos mencionados e imprime los siguientes resultados:

posición 1 posición30 posición 41

0.678909E⊅03⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅⊅12345⊅⊅⊅⊅⊅⊅321

⊅0.87654E⊅03

F. Otras formas de READ Y WRITE

a) Uso de arreglos de códigos de formato

En las proposiciones READ y WRITE es posible utilizar, en vez de una proposición FORMAT, un arreglo con códigos de formato. El nombre de dicho arreglo reemplaza, en la estructura de las proposiciones READ y WRITE, al número de identificación de la proposición FORMAT.

Deben cumplirse las siguientes normas:

1) La información que se guarda en el arreglo debe ser idéntica a la que contiene la proposición FORMAT en su argumento, esto es, se elimina solamente el número de identificación y la palabra clave FORMAT.

2) Debe utilizarse un arreglo, aún cuando éste tenga que tener un sólo elemento.

3) Si el argumento almacenado contiene literales y dentro de alguno de éstos hay doble apóstrofo, se deberá utilizar el argumento sólo en salida, en caso contrario debe ocuparse el código de formato H.

Debe tenerse en cuenta que cada carácter ocupa un byte, sin embargo,es conveniente especificar un arreglo con mayor capacidad que la necesaria para guardar todos los caracteres del argumento, de tal manera que si se cambia este último no sea necesario tener que modificar las dimensiones del arreglo cada vez.

Ejemplo 52:

El programa siguiente almacena códigos de formato en un arreglo y utiliza éste para entrada y salida.

```
C EJEMPLØ 52.
C USØ DE ARREGLØ DE CØDIGØS
      DIMENSIØN FMT(3)
      READ (1,50) FMT
      READ (1,FMT) L,A,B
      X = (A + B)**L
      Y = (A - B)**L
      M = L * 2
      WRITE(3,FMT) M,X,Y
      STØP
   50 FØRMAT (3A4)
      END
```

Los datos que lee el programa son:

(I4,2F7.2)

ØØØ2ØØØ10.5ØØØI1.1

y se obtienen los siguientes resultados:

ØØ4Ø466.56ØØØ0.36

b) Proposición NAMELIST

Es una proposición de especificación, que permite utilizar las proposiciones READ y WRITE sin especificar lista de entrada/salida. Para lograr esto se declaran con NAMELIST nombres de listas de entrada/salida y a dichos nombres se refieren las proposiciones READ y WRITE.

1) Estructura de la proposición

NAMELIST /$X_1$/$lista_1$/$X_2$/$lista_2$/,...,/$X_n$/$lista_n$

donde:

los $X_i$: representan nombres de lista

las $listas_i$ : son listas de entrada/salida

2) Función.

Se asignan nombres simbólicos a listas de entrada/salida. Estos nombres no deben ser los mismos de variables o arreglos y deben aparecer encerrados entre operadores de división. Los de variables o arreglos pueden pertenecer a más de una lista. Los declarados en la proposición NAMELIST deben utilizarse sólo en proposiciones en entrada/salida.

3) Estructura de los datos de entrada para NAMELIST

Los datos deben tener un formato especial para ser leidos con nombres declarados en la proposición NAMELIST

i) El primer carácter en cada registro que se va a leer debe ser blanco.

ii) El segundo carácter en el primer registro de un grupo de registros de datos debe ser ε (epsilon), seguido inmediatamente por el nombre declarado en NAMELIST. Este nombre no debe contener blancos y debe ser seguido por un blanco.

iii) El nombre debe ser seguido, después del blanco, por los ítems de datos, separados entre si por coma, opcional despúes del último ítem de datos.

iv) El término del grupo de datos se señala mediante

ε END

v) El ítem de datos tiene la estructura siguiente:

nombre simbólico = constante (s)

donde:

nombre simbólico es el nombre de una variable, con o sin subíndices, o nombre de arreglo

constante (s) puede ser de cualquier tipo. Si es lógica, puede tener la forma T y F o .TRUE. y .FALSE.. Si son varias constantes que definen a un nombre de arreglo, deben ser, en cantidad, menor o igual al número de elementos del arreglo. Si la misma constante se repite en forma sucesiva, puede adoptarse la notación i*constante, siendo i el número de veces que la constante se repite.

vi) Los nombres simbólicos que aparecen en los datos de entrada deben estar declarados como parte de al menos una lista en la proposición NAMELIST; sin embargo, el orden que ellos tengan es arbitrario.

vii) En las listas de la proposición NAMELIST no puede haber nombres que sean parámetros formales (ver "Subprogramas").

viii) Los blancos que figuren después de enteros y exponentes son tratados como ceros.

ix) Si en la primera tarjeta no encuentra el nombre simbólico con el cual se está leyendo, lo sigue buscando en los grupos NAMELIST siguientes.

4) Estructura de los datos de salida

Los datos se escriben conservando la estructura que tendrían al ser leidos a través de la proposición NAMELIST. Los campos de salida están diseñados para contener todos los dígitos significativos del dato. Los arreglos son escritos por columna.

Ejemplo 53:

El programa que figura a continuación:

```
C EJEMPLØ 53.
C USØ DE PRØPØSICIØN NAMELIST
      NAMELIST /NØM1/L,A,B/NØM2/M,X,Y
      READ (1,NØM1)
      X = (A + B)**L
      Y = (A - B)**L
      M = L * 2
      WRITE(3,NØM2)
      STØP
      END
```

lee los siguientes grupos de datos, en procesos separados

```
&NØM1 A=10.5,L=2,B=11.1&END        proceso 1
                                 2 líneas en blanco
&NØM1 A=10.5                       proceso 2
L=2,B=11.1,&END
```

y en ambos se obtienen los mismos resultados que se indican enseguida:

```
ε NØM2
M=          4,X=  466.55957    ,Y= 0.35999930
ε END
```

Obsérvese que se imprime ε en vez de &. Sin embargo, la configuración o representación interna es la misma.

Ejemplo 54:

Se tienen los siguientes datos:

```
&NØM1 A=8.5,B=1.5,L=2&END          1a tarjeta
&NØM2 C=5.5,D=4.5,I=3&END          2a tarjeta
```

El programa que figura a continuación lee la primera vez con el nombre simbólico NØM2, que no se encuentra en la primera tarjeta. Se salta la primera y consulta en la segunda si está el nombre buscado. El proceso se realiza en forma correcta; sin embargo, el primer grupo NAMELIST se pierde.

```
C EJEMPLØ 54.
C USØ DE PRØPØSICIØN NAMELIST
      NAMELIST /NØM1/L,A,B/NØM2/I,C,D/NØM3/M,X,Y
      READ(1,NØM2)
      X = (C+D)**I
      Y = (C-D)**I
      M = I*2
    1 WRITE(3,NØM3)
      READ(1,NØM1,END=2)
      X = (A+B)**L
      Y = (A-B)**L
      M = L*2
      GØ TØ 1
    2 STØP
      END
```

Los resultados que se obtienen son:

```
ε NØM3
M=          6,X=  1000.0000     ,Y=  1.0000000
ε END
```

G. <u>Otras proposiciones secuenciales de entrada/salida</u>

a.) Proposición END FILE

i) Estructura de la proposición

END FILE a

donde:

a: es una constante entera sin signo o variable entera sin signo y sin subíndices, que representa un número de referencia de conjunto de datos.

ii) Función. Se define el final o término del conjunto de datos asociado con a. Una proposición WRITE después de END FILE define el comienzo de un nuevo conjunto de datos.

b) Proposición REWIND

i) Estructura de la proposición

REWIND a

donde:

a es una constante entera sin signo o variable entera sin signo y sin subíndices que representa un número de referencia de conjunto de datos.

ii) Función. Causa que una proposición READ o una proposición WRITE, inmediatamente posterior, se refieran al primer registro del primer conjunto de datos asociado con a.

c) Proposición BACKSPACE

i) Estructura de la proposición

BACKSPACE a

donde:

a es una constante entera sin signo o variable entera sin signo y sin subíndices, que representa un número de referencia de conjunto de datos.

ii) Función. Causa que se efectúe el retroceso de un registro lógico en el conjunto de datos asociado con a. Si el conjunto de datos estaba en su comienzo al darse la orden, ésta no tiene efecto.

Al término de un archivo deben especificarse dos proposiciones BACKSPACE si se desea recuperar el último registro lógico grabado.

Ejemplo 55:

```
C EJEMPLØ 55.
C USØ DE LAS PRØPØSICIØNES
C END FILE,REWIND Y BACKSPACE
      DIMENSIØN A(20),B(20),C(20)
      N = 8
    1 READ (1,51) A,B,C
      WRITE(N,51) A,B,C
      IF(C(1).NE.0.) GØ TØ 1
      END FILE N
```

(Continúa)

(Continuación)

```
2 READ (1,51) A,B,C
  WRITE(N,51) A,B,C
  IF(C(1).NE.0.) GØ TØ 2
  END FILE N
  REWIND N
3 READ (N,51,END=4) A,B,C
  WRITE(3,52) A,B,C
  GØ TØ 3
4 READ (N,51,END=5) A,B,C
  WRITE(3,52) A,B,C
  GØ TØ 4
5 READ (N,51,END=6)
  GØ TØ 5
6 BACKSPACE N
  BACKSPACE N
  READ (N,51,END=7) A,B,C
  WRITE(3,52) A,B,C
7 STØP
51 FØRMAT(20F4.1)
52 FØRMAT(' ',20F6.1)
   END
```

Con este programa se graban dos archivos en un carrete de cinta magnética, cada uno de los cuales está formado por dos registros físicos y cada registro físico por tres registros lógicos. El término de cada archivo se obtiene con la proposición END FILE que causa la grabación de una marca de fin de archivo. La proposición REWIND rebobina la cinta al punto de carga. Las dos proposiciones BACKSPACE posicionan la cinta en el último registro lógico grabado.

## H. Problemas propuestos

a) ¿Cuántas tarjetas se leen con la siguiente serie de proposiciones:

```
      READ (1,3)(A(I),I=1,5), (B(I),I=1,7)
3     FØRMAT (2F8.3,3(F5.2/F4.1))     ?
```

b) Se tienen los siguientes datos:

A = 2.5 , B = -37.72 , C = -732.5 , I = 876

i) Ordenarlos en tarjetas y leerlos

ii) ¿Cómo quedan los resultados con:

```
      WRITE (3,2) A,B,I,C
    2 FØRMAT (2F8.3,I5)
```

?

c) Se tiene una tarjeta perforada como se indica a continuación:

```
0203532.4729399087
```

se pide la impresión, luego de ejecutar el siguiente programa:

```
      READ (1,3) I,AI,BJ,L,DATØ
      WRITE(3,5) AI,BJ,DATØ,L,I
      STØP
    1 FØRMAT (3F9.2/(I4))
    3 FØRMAT (I4,F6.4,F3.2,I3,F7.2)
    5 FØRMAT (3F5.3/(I3))
      END
```

d) Se tiene una tarjeta con los siguientes caracteres

```
ABXDGFØERRTSRANQPYØR
```

se piden los resultados luego de procesar el siguiente programa:

```
      READ (1,3) A,B,C,D
      WRITE(3,5) A,C,B,D
      STØP
    3 FØRMAT (4A5)
    5 FØRMAT (A1,A3,A4,A2)
      END
```

## 4. Proposiciones de Especificación

Las proposiciones de especificación proporcionan al compilador información acerca de la naturaleza de los datos, como también información que le permite asignar a dichos datos ubicaciones en memoria o reservar memoria para resultados.

Toda proposición de especificación debe preceder a la primera proposición ejecutable del programa fuente.

A. Proposición DIMENSION

i) Estructura de la proposición

DIMENSIØN $a_1(k_1)$, $a_2(k_2)$, ..., $a_n(k_n)$

donde:

los $a_i$ son nombres de arreglos

los $k_i$ representan las dimensiones del arreglo. Cada $k_i$ está compuesto de una a siete constantes enteras, sin signo, separadas entre sí por coma cuando hay más de una. Cada constante representa el valor máximo que puede tener el subíndice respectivo dentro del arreglo. Cada $k_i$ puede contener variables enteras, de longitud 4 bytes, sólo cuando la proposición DIMENSION en la cual ellas aparecen forma parte de un SUBPROGRAMA (véase el capítulo Subprogramas "Dimensiones en tiempo de ejecución").

ii) Función. Permite asignar memoria a los arreglos que se utilizan dentro del programa fuente.

Ejemplo 56:

```
C EJEMPLØ 56.
C USØ DE PRØPØSICIØN DIMENSIØN
      DIMENSIØN A(10,10)
      S = -3.
      DØ 10 I = 1,4
      S = S + 4.
      DØ 10 J = 2,4
      A(I,1) = S
   10 A(I,J) = A(I,J-1) + 1.
      WRITE(3,51)((A(I,J),J=1,4),I=1,4)
      STØP
   51 FØRMAT(4(F6.2,5X))
      END
```

Los resultados que se obtienen con el programa anterior son los siguientes:

| | | | |
|---|---|---|---|
| 1.00 | 2.00 | 3.00 | 4.00 |
| 5.00 | 6.00 | 7.00 | 8.00 |
| 9.00 | 10.00 | 11.00 | 12.00 |
| 13.00 | 14.00 | 15.00 | 16.00 |

B. Proposiciones de tipo

Existen dos clases de proposiciones de tipo: la proposición IMPLICIT y las proposiciones de especificación explícitas.

a) Proposición IMPLICIT

Debe ser la primera proposición en un programa principal y la segunda en un subprograma y no puede haber más de una en ninguno de los dos.

i) Estructura de la proposición

IMPLICIT $tipo_1 *s_1(a_{11},a_{12},\ldots),\ldots, tipo_n *s_n(a_{n1}, a_{n2}, \ldots)$

donde:

los $tipo_i$ pueden ser alguna de las siguientes palabras claves:

INTEGER, REAL, LOGICAL

los $s_i$ son constantes enteras sin signo, opcionales y representan alguna de las longitudes permitidas para el tipo al cual están asociadas. Si no se coloca, debe eliminarse el asterisco que le precede en la estructura.

los $a_{ij}$ representan un carácter o un rango de caracteres alfabéticos (A, B, ...,Z,$). El rango se representa a su vez por el primero y por el último carácter, separados entre sí por un operador de resta y manteniendo el mismo orden en que están en el conjunto de caracteres alfabéticos, esto es, el rango de C a J se representa por (C-J) y no (J-C).

ii) Función. Se especifica el tipo y longitud de todas las variables, arreglos y funciones del usuario (véanse los "Subprogramas"), cuyos nombres empiecen con una letra en particular.

Tiene prioridad sobre la declaración predefinida de tipo.

Ejemplo 57:

1) IMPLICIT INTEGER (A-F), REAL (I-N)

Todas las variables cuyos nombres empiecen con las letras incluidas en el rango A a F son declaradas enteras y aquéllas cuyos nombres empiecen con las letras del rango I a N son declaradas reales de simple precisión.

2) IMPLICIT INTEGER * 2 (A-E, Ø-$), REAL*8 (J,K)

Todas las variable cuyos nombres empiecen con las letras de los rangos A a E y 0 a $ son declaradas enteras de longitud dos bytes y aquéllas cuyos nombres empiecen con las letras J y K son declaradas reales de doble precisión.

b) Proposiciones de especificación de tipo explícitas

i) Estructura de las proposiciones

Tipo*S $a_1$*$s_1$($k_1$)/$X_1$/,$a_2$*$s_2$($k_2$)/$X_2$/,...,$a_n$*$s_n$($k_n$)/$X_n$/

donde:

Tipo es INTEGER, REAL, LOGICAL

S y $s_i$ son constantes enteras sin signo, opcionales y representan alguna de las longitudes permitidas para el tipo al cual están asociadas. Si no se coloca, el asterisco que le precede en la estructura debe eliminarse.

los $a_i$ son nombres de variables, arreglos o funciones(véanse los "Subprogramas").

los $k_i$ son opcionales y representan las dimensiones del arreglo (véase "Proposición DIMENSION").

los $x_i$ son opcionales y representan valores de datos iniciales. Si no se coloca, se omiten los operadores de división //

ii) Función. Declara el tipo de una variable, arreglo o resultado de una función por su nombre, independiente del primer carácter de éste. Se puede, además, dar la dimensión de los arreglos cuyo tipo se declara (en ese caso, no deben aparecer en la proposición DIMENSION).

Si se desea asignar valores iniciales a las variables o arreglos que se estén declarando, o ambas cosas a la vez, esos valores se encierran entre operadores de división inmediatamente a continuación de las variables o arreglos que se inicialicen. Este se efectúa para el arreglo o variable inmediatamente precedente. Debe haber correspondencia entre el tipo del dato y la variable respectiva, exceptuando el caso de constantes literales o hexadecimales. Si los valores que se desea asignar se repiten en forma sucesiva, se puede utilizar la notación i * constante, en que i indica el número de veces que se repite la constante. Para inicializar arreglos, éstos deben estar dimensionados en la misma proposición o en proposiciones DIMENSION o COMMON precedentes. No se pueden asignar valores iniciales a nombres de función.

La declaración de tipo mediante proposiciones de especificación explícitas tiene prioridad sobre la proposición IMPLICIT y sobre la declaración predefinida.

Ejemplo 58:

Se tienen los siguientes datos:

```
ABCDEFGHIJKLMNØPQRSTUVWXYZ$
```

El programa que figura a continuación:

```
C EJEMPLØ 58.
C USØ DE LAS PRØPØSICIØNES DE ESPECIFICACIØN
C INTEGER,REAL,LØGICAL
      INTEGER*2 ALFA,BETA,GAMA*4/2555/
      INTEGER A/'DATØ'/,B/ZFF/
      REAL*8 C(2,3)/5*1.,2./,D*4/Z42FF0000/
      LØGICAL*1 L,M,N/ØF/
      LØGICAL Ø,P,Q/T/,R/.FALSE./
      READ (1,51) ALFA,BETA,L,M,Ø,P
      WRITE(3,52) ALFA,BETA,GAMA,A,B,C,D
      WRITE(3,53) L,M,N,Ø,P,Q,R
      STØP
   51 FØRMAT(2A4,2A3,2A6)
   52 FØRMAT(' ',2A8,I6,A6,I6/(' ',3D11.3))
   53 FØRMAT(' ',2A4,L4,2A8,2L4)
      END
```

lee los datos mencionados y entrega los siguientes resultados impresos:

```
ØØØØØØCDØØØØØØGHØØ2555ØØDATØØØØ255          1a línea
ØØ0.100DØ01ØØ0.100DØ01ØØ0.100DØ01           2a línea
ØØ0.100DØ01ØØ0.100DØ01ØØ0.200DØ01           3a línea
ØØ0.255EØ03                                 4a línea
ØØØKØØØNØØØFØØØØQRSTØØØØWXYZØØØTØØØF       5a línea
```

c) <u>Proposición DOUBLE PRECISION</u>

i) Estructura de la proposición

DOUBLE PRECISION $a_1(k_1)$, $a_2(k_2)$,..., $a_n(k_n)$

donde:

los $a_i$ representan nombres de variables, arreglos o funciones (ver "Subprogramas")

los $k_i$ son opcionales y representan las dimensiones del arreglo (ver "Proposición DIMENSION")

ii) Función. Se declara en forma explícita que las variables, arreglos o resultados de funciones que aparecen en la proposición son reales de doble precisión.

Ejemplo 59:

i) DOUBLE PRECISION A,X,Z

ii) DOUBLE PRECISION I,J(20),B(10,10)

iii) DOUBLE PRECISION C(5,4,5),N,Y

C. Problemas propuestos

a) Se pide el valor de A y de S al final del programa siguiente

```
      INTEGER A
      S = 0.
      A = 5
    5 DØ 10 I=1,A
   10 S = S + A*I
      A = A + 2
      IF(A - 10)5,5,6
    6 IF(S - 100.)7,7,8
    7 A = 1
      GØ TØ 9
    8 A = 2
    9 GØ TØ (11,12),A
   11 S = 0.
   12 STØP
      END
```

b) Programar el cálculo de la suma de los cuadrados de los 100 elementos de una lista

$$S = \sum_{i=1}^{100} a_i^2$$

c) Se tiene un arreglo bidimensional con 20 renglones y 10 columnas. Programar el cálculo de la suma de los elementos

d) Se tienen dos matrices A(M,N) y B(M,N). Programar

$$C = A + B$$

e) Se tienen dos matrices A(i,j) y B(j,k). Se pide programar el cálculo de:

$$C(i,k) = A(i,j) * B(j,k)$$

f) Se tienen dos listas A y B con 50 elementos cada una. Programar el cálculo de los elementos de un arreglo C, como sigue:

si $a_i \neq 0$ $C_i$ es la suma de los elementos de B, menos $b_i$

si $a_i = 0$ $C_i$ es la suma de los elementos de A, hasta $a_i$

g) Se tienen dos listas A y B de n elementos cada una (N< 500). Se pide una lista C de N elementos formados de la siguiente manera:

$$C_i = \sqrt{\frac{|b_i + a_i|}{1 + |a_i|}}$$

para el cálculo de la raíz usar la fórmula iterativa

$$X_{i+1} = \frac{1}{2}\left(X_i + \frac{A}{X_i}\right)$$

h) Tabular la siguiente función:

$$Z = \frac{X^2 - 2XY - Y^2}{2X - 3Y}$$

para

X = -1.(0.1)1.

Y = -1.(0.1)1.

Si el denominador es menor o igual a 0.01 suponerlo 0 y hacer Z = 10**75

i) Tabular la siguiente función:

$$Y = AX^2 + B$$

para X = 0.1(0.1)2.
A = -5.(1.)5.
B = 0. (0.5)10.

j) Se tiene una lista de valores, LISTA (1000). Se pide: el cuadrado de los elementos, el cubo de los elementos y el valor recíproco de ellos. Además la suma de los cuadrados, de los cubos y de los recíprocos.

k) Dadas 100 tarjetas, en cada una de las cuales se tienen perforados tres valores (separados entre sí por blanco), escribir un programa para leer las tarjetas y colocar los valores en listas A,B y C respectivamente.

l) Escribir un programa que lea 100 valores. Los 50 primeros se almacenan en un arreglo A y los restantes en un arreglo B.

m) Se tiene en memoria una lista J de cien elementos. Escribir los valores que corresponden a elementos de índice impar.

n) Se pide la salida del siguiente programa:

```
      DIMENSIØN A(12)
      S = 1.
      DØ 10 I=1,10,3
      S = S * I
      A(I) = S
      A(I+1) = 2 * S
   10 A(I+2) = S + 5
      WRITE(3,11)(A(I),I=1,12,2),(A(I),I=2,12,2)
   11 FØRMAT(F7.2,F5.0,(F7.2,2F6.1))
      STØP
      END
```

ñ) Escribir un programa que imprima los cien elementos de una matriz A, cuatro por línea, con la organización que se indica:

A(1) = A(2) = A(3) = A(4) =
A(5) = A(6) = etc.

o) Se pide indicar la salida de resultados de los siguientes programas:

i)

```
      DIMENSIØN A(100)
      S = 0.
      DØ 10 I=1,6
      S = S + 1.
   10 WRITE (3,1) S
    1 FØRMAT(4(F6.2,5X))
      STØP
      END
```

ii)

```
      DIMENSIØN A(100)
      A(1) = 1
      DØ 10 I=1,14
   10 A(I+1) = A(I)+1.
      WRITE(3,1)(A(I),A(I+1),A(I+2),I=1,3)
    1 FØRMAT(3(F6.2,5X))
      STØP
      END
```

iii)

```
      DIMENSIØN A(100)
      S = 0.
      DØ 10 I =1,16
      S = S + 1.
   10 A(I) = S
      WRITE(3,1)(A(I),I=1,16)
    1 FØRMAT(4(F6.2,5X))
      STØP
      END
```

iv)

```
      DIMENSIØN A(10,10)
      S = -3.
      DØ 10 I=1,4
      DØ 10 J=2,4
      S = S + 4.
      A(I,1) = S
   10 A(I,J) = A(I,J-1)+1.
      WRITE(3,1)((A(I,J),J=1,4),I=1,4)
    1 FØRMAT(4(F6.2,5X))
      STØP
      END
```

v)

```
      DIMENSIØN A(10,10)
      A(1,1) = 1.
      DØ 10 I=1,4
      DØ 10 J=2,4
      A(I+1,1)=A(I,1)+4.
   10 A(I,J)=A(I,J-1)+1.
      WRITE(3,1)((A(I,J),J=1,4),I=1,4)
    1 FORMAT(4(f6.2,5X))
      STOP
      END
```

p) Se pide la impresión que entrega el siguiente programa:

```
      DIMENSIØN A(10),B(10)
      S = 0.
      A(1)=S
      A(2)=S + 1.
      DØ 10 I=3,10
      S = S + 1.
   10 A(I)=A(I-1)*S-A(I-2)*(S+1.)
      K = 0
      DØ 20 I=1,10
      IF(A(I)-5)20,20,3
    3 K = K + 1
```

(Continúa)

(Continuación)

```
      B(K) = A(I)
   20 CØNTINUE
      WRITE(3,2)(B(J),J=1,K)
    2 FØRMAT(T10,F7.2,T30,F9.5,T70,F5.1)
      STØP
      END
```

q) Se pide la impresión que entrega el siguiente programa:

```
      DIMENSIØN B(10),A(10,10)
      DØ 10 I=1,5
   10 B(I) = I
      DØ 15 I=1,5
      A(I,1) = B(I)
      A(I,2) = A(I,1) + 1.
      DØ 15 J=3,5
      A(I,J) = 0.
      L = J - 1
      DØ 15 K=1,L
   15 A(I,J) = A(I,J) + A(I,K)
      WRITE(3,3)((A(L,K),L=1,5),K=1,5)
      STØP
    3 FØRMAT(5F10.2/(5F9.3))
      END
```

r) Los resultados de una encuesta fueron perforados con la siguiente disposición en tarjetas:

| | |
|---|---|
| Nombre de la encuestada | columnas 1 a 24 |
| Estado Civil | columna 25 (0 = soltera,1 = casada y otros) |
| Nacionalidad | columnas 26 a 29 |
| Número de hijos | columnas 30 y 31 |

Se desea saber:

El promedio de hijos

El porcentaje de madres solteras

El porcentaje de madres con más de tres hijos

El porcentaje de hijos naturales

Se ha colocado una última tarjeta que tiene perforado un nueve en columna 25 para utilizarla como fin de archivo.

## 5. Subprogramas

En programación se presenta con mucha frecuencia la necesidad de tener que realizar repetidas veces partes de un programa, que corresponden a cálculos en los que la única variación se efectúa en los valores que toman las variables. Por ejemplo, en el cálculo siguiente:

```
PDX=((((A5*X+A4)*X+A3)*X+A2)*X+A1)*X+A0
```

se pueden tener los coeficiente constantes y realizar el cálculo de PDX para distintos valores de X o mantener constante X y efectuar el cálculo para distintos juegos de coeficientes. En ambos casos, el tener que detallar toda la proposición de asignación cada vez que se necesite un resultado, constituye una pérdida de tiempo que será mayor cuanto mayor sea la complejidad del cálculo o el número de proposiciones que haya que repetir.

Se evita esta situación con los subprogramas, denominados también rutinas o subrutinas, que pueden ser "llamados" por el programa principal o monitor cada vez que se necesite el resultado deseado.

Es importante tener presente que el uso de subprogramas permite darle modularidad a un programa, esto es, estructurarlo a base de módulos, cada uno de los cuales podría ser programado por personas distintas a base de: información que se entrega al módulo, proceso de cálculo y resultado(s) que debe entregar. Es necesario tener claro que el uso en sí de subprogramas no constituye modularidad dado que los subprogramas deben cumplir con normas precisas de construcción y funciones a realizar, dentro de la estructura total del programa.

Se tienen dos clases de subprogramas: subprograma FUNCTION y subprograma SUBROUTINE. Se ven además en este capítulo, las "funciones de proposición" y las funciones estándar.

### A. Funciones de proposición

Las funciones de proposición se definen (declaran) y son "llamadas" dentro de la misma unidad de programa.

i) Estructura de la función de proposición

nombre $(a_1, a_2, \ldots, a_n)$ = expresión

donde:

nombre: es la identificación de la función

los $a_i$: son variables sin signo, sin subíndices, distintos entre sí llamados parámetros formales (argumentos vacíos). Debe haber al menos un parámetro formal

expresión: es cualquier expresión aritmética o lógica que no contenga variables con subíndice. Si dentro de la expresión aparece una función de proposición, ésta debe estar definida previamente.

ii) Función. Se establece un procedimiento de cálculo (la expresión) en el cual intervienen los parámetros formales. Dichos parámetros formales son reemplazados, uno a uno, por parámetros actuales o reales, cuando se realiza la llamada de la función.

La llamada de la función se ejecuta al aparecer el nombre de la función en una proposición de asignación, seguido de los parámetros actuales encerrados entre paréntesis.

Los parámetros actuales deben corresponder en tipo, número y orden con los parámetros formales.

Para la declaración del tipo de la función se aplican las mismas reglas que para la declaración del tipo de variables.

Una vez que los parámetros actuales reemplazan a los formales en el procedimiento de cálculo, se evalúa la expresión y el valor obtenido reemplaza a su vez la llamada de la función en la proposición de asignación.

Los nombres de los parámetros formales pueden aparecer en varias funciones de proposición y pueden asimismo ser utilizados como nombres de variables pues constituyen en la práctica elementos distintos.

La función de proposición debe aparecer antes de cualquier proposición ejecutable. Si en ella aparece en la parte expresión una llamada de otra función de proposición, esta última debe estar previamente definida. La función de proposición no puede llamarse a sí misma.

Ejemplo 60:

a) Definiciones válidas

i) SUMC(A,B,C) = A*A+B*B+C*C

ii) POL(A1,A2,A3) = (A1*X+A2)*X+A3

iii) PROM(X,Y,Z) = (X+Y+Z)/3.

iv) VLOG(A,B,C) = A.AND.B.OR.C

b) Definiciones no válidas

i) ALFA(3.,X) = A*X+3. una constante aparece como parámetro formal

ii) POL(A(1),A(2)) = A(1)*X+A(2) aparecen variables con subíndices como parámetros formales y además en la expresión

iii) POLIN(X) = A(1)*X+A(2) variables con subíndices en la expresión

iv) TAB(X,Y) = X**2+Y*TAB(A,B) llamada a sí misma de la función

Ejemplo 61:

En una tarjeta se tienen perforados los siguientes datos:

```
1  15-1  2  25 3 -35-3  1 -2  25  15
```

que son leidos por el programa que figura a continuación:

```
C EJEMPLØ 61.
C USØ DE FUNCIØN DE PRØPØSICIØN
      DIMENSIØN X(8),Z(7)
      VALØR (XZ) = ((A3*XZ+A2)*XZ+A1)*XZ+A0
      READ (1,51) X,A3,A2,A1,A0
      Z(1) = VALØR(X(1))**2 + 5.
      Z(2) = .5 + VALØR (X(2)**2) * 2.
      Z(3) = 3. * VALØR (X(3)+1.5)
      DØ 15 I=4,7
   15 Z(I) = VALØR (X(I))*VALOR(X(I+1))
      WRITE(3,52) Z
      STØP
   51 FØRMAT(12F3.1)
   52 FØRMAT(' ',3(F10.2,2X))
      END
```

El programa entrega los resultados que se indican:

```
ØØØØØ14.00ØØØØØØØ17.28ØØØØØØØØ7.13
ØØØØ70.69ØØØØØØ195.75ØØØØ-1343.25
ØØ3805.88
```

Ejemplo 62:

```
C EJEMPLØ 62.
C USØ DE FUNCIØN DE PRØPØSICIØN
      ALFA(Y) = (A*Y + B)*Y + A*B
      BETA(X) = (3*X + A)/(3*X + B) + ALFA(X)
      READ (1,51) A,B
      Y = ALFA(.5)
      Z = BETA(.5)
      WRITE(3,52) Y,Z
      STØP
   51 FØRMAT(2F3.1)
   52 FØRMAT(' ',2F10.3)
      END
```

El programa anterior lee los siguientes datos

2 3

y entrega los resultados que se indican a continuación:

ØØØØØ8.000ØØØØØ8.778

Ejemplo 63:

```
C EJEMPLØ 63.
C USØ DE FUNCIØN DE PRØPØSICIØN
      LØGICAL*1 L,M,U,V,W,X,Y,Z,A,B,C,VALØG
      VALØG(A,B,C)=.NØT.A.AND.B.ØR.C
      READ (1,51) U,V,W,X,Y,Z
      IF(VALØG(U,V,W)) GØ TØ 1
      L = .FALSE.
      WRITE(3,52) L
    1 IF(VALØG(X,Y,Z)) GØ TØ 3
      M = .FALSE.
    2 WRITE(3,52) M
      STØP
```

(Continúa)

(Continuación)

```
 3 M = .TRUE.
   GØ TØ 2
51 FØRMAT (6L2)
52 FØRMAT (' ',2(L3,2X))
   END
```

El programa anterior lee los siguientes datos:

F T F T F T

y entrega el resultado que se indica a continuación:

T

B. Funciones estándar

Corresponden a procedimientos de cálculo de uso frecuente y que, por este motivo, han sido incorporados al lenguaje FORTRAN. Esto significa que es posible llamar a las funciones estándar en la misma forma en que se efectúa el llamado de las funciones de proposición, esto es, en una proposición de asignación se coloca el nombre de la función estándar, y a continuación encerrados entre paréntesis se especifican el o los argumentos.

A continuación se entrega una lista de las funciones estándar de uso más común. En el Apéndice A aparece una lista completa.

| Función | Nombre | Definición |
|---|---|---|
| Logaritmo natural | ALOG<br>DLOG | $y = \log_e x$ ó<br>$y = \ln x$ |
| Logaritmo decimal | ALOG10<br>DLOG10 | $y = \log_{10} x$ |
| Exponencial | EXP<br>DEXP | $y = e^x$ |
| Raíz cuadrada | SQRT<br>DSQRT | $y = \sqrt{x}$ ó<br>$y = x^{1/2}$ |
| Seno | SIN<br>DSIN | $y = \sin x$ |

| Función | Nombre | Definición |
|---|---|---|
| Coseno | COS<br>DCOS | y = cos x |
| Tangente | TAN<br>DTAN | y = tan x |
| Cotangente | COTAN<br>DCOTAN | y = cotan x |
| Valor absoluto | IABS<br>ABS<br>DABS | y = \|x\| |

Nota: Los nombres que empiezan con D corresponden a doble precisión, los otros a precisión simple, excepto IABS, que es el valor absoluto de un entero. El tipo del argumento debe corresponder a la precisión que se desea. Los argumentos de las funciones trigonométricas deben darse en radianes [-].

Ejemplo 64:

Programar el cálculo de:

$$y = e^{\log_e \sqrt{\frac{\text{sen } x + \cos x}{|A + B|}}}$$

Con la ayuda de las funciones estándar, la solución se obtiene fácilmente:

```
Y = EXP(ALOG(SQRT((SIN(X)+COS(X))/ABS(A+B))))
```

C. <u>Subprograma FUNCTION</u>

Consiste en la proposición FUNCTION seguida de otras proposiciones entre las que debe aparecer, al menos una vez, una proposición de asignación en que la variable que figura a la izquierda del símbolo de asignación sea el nombre del subprograma, y una proposición RETURN. Al final de todas debe estar la proposición END.

La llamada del subprograma FUNCTION se efectúa en la misma forma que la llamada de una función de proposición.

El subprograma FUNCTION constituye un módulo independiente del programa principal, por lo cual se pueden definir en él: variables, arreglos y números de identificación de proposición, iguales a los definidos en el programa principal o en otros subprogramas. No puede contener proposiciones SUBROUTINE u otra proposición FUNCTION, ni llamarse a sí mismo.

a) Proposición FUNCTION

i) Estructura de la proposición

Tipo FUNCTION nombre * s $(a_1, a_2, \ldots, a_n)$

donde:

Tipo puede ser: INTEGER, REAL, DOUBLE PRECISION o LOGICAL. Es optativo colocarlo.

nombre: es el identificador de la función

*s: representa una de las longitudes permitidas para el tipo asociado. Puede ser colocada o no, siempre que se haya especificado Tipo.

los $a_i$: son parámetros formales. Debe haber al menos uno.

ii) Función. Las proposiciones que siguen a la proposición FUNCTION constituyen una declaración ó definición de un procedimiento de cálculo, en el cual deben figurar los parámetros formales, que son reemplazados por los actuales cuando el subprograma es llamado (véase "Parámetros actuales en Subprogramas").

Normalmente se entrega un resultado a través del nombre del subprograma. Dicho nombre debe aparecer, al menos una vez, a la izquierda del símbolo de definición en una proposición de asignación. Es posible entregar más de un resultado haciendo uso de parámetros formales, los cuales tendrán que figurar en proposiciones de asignación igual que el nombre del subprograma.

La relación entre parámetros formales y actuales es la misma que en las funciones de proposición.

Si no se especifica "Tipo", éste se indica en alguna de las otras formas posibles, esto es: declaración predefinida, declaración explícita o por medio de la proposición IMPLICIT. Si se utiliza esta última dentro del subprograma, debe estar inmediatamente a continuación de la proposición FUNCTION. Cuando no se hace uso de la declaración predefinida, debe declararse el tipo también en el programa llamador.

b) Proposición RETURN

i) Estructura de la proposición

RETURN

RETURN i

donde:

i es una variable sin subíndices, sin signo o constante entera sin signo. Se utiliza solamente en subprogramas SUBROUTINE (véase el capítulo "Proposición RETURN en subprograma SUBROUTINE).

ii) Función. Permite el retorno al programa o subprograma que ha llamado al subprograma en el que está la proposición RETURN considerada.

Indica, entonces, la conclusión lógica del cálculo y retorna el control y al menos un resultado numérico al programa llamador. Puede existir más de una proposición RETURN en un subprograma.

El primer formato se puede utilizar en subprogramas FUNCTION o SUBROUTINE. El segundo formato sólo se puede ocupar en subprogramas SUBROUTINE.

Ejemplo 65:

```
C EJEMPLØ 65.
C USØ DE SUBPRØGRAMA FUNCTIØN
      READ (1,51) X,Y,Z
      ZETA = PRUEBA (X,Y,Z,U)
      WRITE(3,52) X,Y,Z,ZETA,U
      STØP
   51 FØRMAT(3F2.1)
   52 FØRMAT(' DATØS',3F5.1/' RESULTADØS',2F7.2)
      END

      FUNCTIØN PRUEBA(A,B,C,D)
      PRUEBA =(2.*A + 3.*B)/4.*C
      D = PRUEBA ** 2
      RETURN
      END
```

El programa anterior lee los siguientes datos:

453040

y entrega los resultados que figuran a continuación:

DATØSØØ4.5ØØ3.0ØØ4.0

RESULTADØSØØ18.00Ø324.00

Ejemplo 66:

En una tarjeta se tienen perforados los datos que se indican:

5101520253035404550556065707580859095 10

Los datos son leidos por el programa siguiente:

```
C EJEMPLØ 66.
C USØ DE SUBPRØGRAMA FUNCTIØN
      DIMENSIØN A(20)
      READ (1,51)A
      PR = PRØM(A)
      WRITE(3,52)
      WRITE(3,53)A,PR
   51 FØRMAT(20F2.1)
   52 FØRMAT(' MATRIZ A'/)
   53 FØRMAT(' ',10F4.1/' ',10F4.1/' PRØMEDIØ',F7.2)
      STØP
      END
      FUNCTIØN PRØM(X,N)
      DIMENSIØN X(N)
      S = 0.
      DØ 10 I=1,N
   10 S = S + X(I)
      PRØM = S/N
      RETURN
      END
```

que entrega los resultados que figuran a continuación:

```
MATRIZ A

Ø0.5Ø1.0Ø1.5Ø2.0Ø2.5Ø3.0Ø3.5Ø4.0Ø4.5Ø5.0
Ø5.5Ø6.0Ø6.5Ø7.0Ø7.5Ø8.0Ø8.5Ø9.0Ø9.5Ø1.0
PROMEDIØØØØ4.80
```

El el subprograma PROM se puede observan que el arreglo X tiene como dimensión una variable entera y el valor de esa variable es pasado al subprograma en el momento de llamarlo. Mayores detalles se pueden ver en el capítulo "dimensionamiento en tiempo de ejecución (objeto)".

Ejemplo 67:

```
C EJEMPLØ 67.
C USØ DE SUBPRØGRAMA FUNCTIØN
  READ (1,51)D,E,F
  Z = 1.5
  SQ = RAIZ(D,E,F)
  SQ = SQ * Z
  WRITE(3,52)SQ
  STØP
51 FØRMAT(3F4.1)
52 FØRMAT(1HO,F7.3)
  END
  FUNCTIØN RAIZ(A,B,C)
  VALOR(X) = (U*X+V)*X+W
  Z = 0.5
  W = Z*3 + 1.5*A
  V = Z*2 + A*B
  U = Z + A*B/C
  RAIZ = VALØR(Z)+A+B-C
  ZETA = VALØR(Z)
  WRITE(3,52)ZETA
52 FØRMAT(' ',F9.2)
  RETURN
  END
```

El programa anterior lee los datos siguientes:

3.0 1.5-2.0

y entrega los resultados que figuran a continuación:

ØØØØ 8.31

Ø22.219

Ejemplo 68:

```
C EJEMPLØ 68.
C USØ DE SUBPRØGRAMA FUNCTIØN
      REAL*4 LISTA
      INTEGER ALFA,SI,SU,EPSIL
      READ (1,51) A,B,C
      SI = ALFA (A,B,C)
      WRITE(3,52) SI
      SU = EPSIL(A,B,C)
      WRITE(3,52) SU
      SV = LISTA(A,B,C)
      WRITE(3,53) SV
      STØP
   51 FØRMAT(3F2.1)
   52 FØRMAT(' ',I5)
   53 FØRMAT(' ',F8.3)
      END
      FUNCTIØN ALFA(X,Y,Z)
      INTEGER ALFA
      ALFA = X*X + Y*Y + Z*Z
      RETURN
      END
      FUNCTIØN EPSIL (X,Y,Z)
      IMPLICIT INTEGER(E-F)
      EPSIL = X*Y*Z
      RETURN
      END
      FUNCTIØN LISTA (X,Y,Z)
      REAL*4 LISTA
      LISTA = X*Y*Z
      RETURN
      END
```

El programa anterior lee los datos siguientes:

354555

y entrega los resultados que figuran a continuación:

62

86

86.625

Ejemplo 69:

```
C EJEMPLØ 69.
C USØ DE SUBPRØGRAMA FUNCTIØN
      IMPLICIT REAL(M-N)
      INTEGER SS,BETA,SW*2,GUIA*2
      READ (1,51) A,B,C
      SR = NETA (A,B,C)
      WRITE(3,53) SR
      ST = GAMA (A,B,C)
      WRITE(3,53) SI
      SS = BETA (A,B,C)
      WRITE(3,52) SS
      SW = GUIA (A,B,C)
      WRITE(3,52) SW
      STØP
   51 FØRMAT(3F2.1)
   52 FØRMAT(' ',I5)
   53 FØRMAT(' ',F8.3)
      END
      FUNCTIØN NETA(X,Y,Z)
      REAL NETA
      NETA = X*X + Y*Y + Z*Z
      RETURN
      END


      INTEGER FUNCTIØN GUIA*2(X,Y,Z)
      GUIA = (X+Y+Z)/3
      RETURN
      END
```

```
FUNCTIØN GAMA(X,Y,Z)
GAMA =(X+Y+Z)/3.
RETURN
END
INTEGER FUNCTIØN BETA(X,Y,Z)
BETA =(X+y+Z)/3
RETURN
END
```

El programa anterior lee los datos siguientes:

354555

y entrega los resultados que figuran a continuación:

```
ØØ62.750
ØØØ4.500
ØØØØ4
ØØØØ4
```

D. Subprogramas SUBROUTINE

Consisten en la proposición SUBROUTINE seguida de otras proposiciones entre las que debe aparecer, al menos una vez, una proposición RETURN y al final de todas, la proposición END.

En muchos aspectos es similar al subprograma FUNCTION y así constituye también un módulo, independiente del programa principal, con iguales características que aquél, en lo que se refiere a nombres de variables y números de identificación de proposición.

Las normas sobre parámetros formales y actuales son las mismas que en el subprograma FUNCTION, exceptuando el hecho de que en el subprograma SUBROUTINE puede no haber parámetros formales, en cuyo caso se omiten los paréntesis.

Difiere el subprograma SUBROUTINE del FUNCTION en la forma de ser llamado, dado que esta acción se realiza a través de una proposición CALL. Además, no necesariamente retorna un resultado numérico, el procedimiento de cálculo puede consistir en copiar una matriz o en transponerla, etc.

No puede contener proposiciones FUNCTION ni SUBROUTINE y tampoco puede llamarse a si mismo.

a) Proposición SUBROUTINE

i) Estructura de la proposición

SUBROUTINE nombre($a_1$,$a_2$,...$a_n$)

donde:

nombre es el identificador de la subrutina

los $a_i$ son parámetros formales

ii) Función. Las proposiciones que siguen a la proposición SUBROUTINE constituyen una declaración o definición de un procedimiento de cálculo. Si hay parámetros formales, se reemplazan por los parámetros actuales cuando el subprograma es llamado (véase "Parámetros actuales en Subprogramas").

Puede utilizar uno o más parámetros formales para retornar resultados al programa llamador. El nombre de la subrutina no puede aparecer en ninguna otra proposición en el subprograma.

Si se utiliza una proposición IMPLICIT, debe aparecer inmediatamente después de la proposición SUBROUTINE.

b) Proposición CALL

i) Estructura de la proposición

CALL nombre ($a_1$,$a_2$,...,$a_n$)

donde:

nombre: es el identificador de un subprograma SUBROUTINE

los $a_i$: son parámetros actuales

ii) Función. Permite llamar a un subprcgrama SUBROUTINE

Ejemplo 70:

El programa siguiente lee un dato con el cual define a la variable X. Transfiere el dato leido al subprograma PRUEBA a través de una proposición CALL. El subprograma PRUEBA entrega al programa principal tres resultados a través de los parámetros formales P2, P3 y A. Los dos primeros se definen con proposiciones de asignación y el último con una proposición READ. Observar que el parámetro formal A es distinto del parámetro actual A.

```
C EJEMPLØ 70.
C USØ DE SUBPRØGRAMA SUBRØUTINE
      READ (1,51) X
      CALL PRUEBA(X,A,B,C)
      SUMA = A + B + C
      WRITE(3,52) SUMA
      STØP
   51 FØRMAT(F3.1)
   52 FØRMAT(' ',F7.3)
      END
      SUBRØUTINE PRUEBA(X,P2,P3,A)
      P2 = 1.5*X**2 - 0.5
      P3 = 2.5*X**3 - 1.5*X
      READ(1,51) A
      RETURN
   51 FØRMAT (F4.0)
      END
```

Datos leidos:

```
.5
20.
```

Resultado obtenido:

```
Ø19.438
```

Ejemplo 71:

```
C EJEMPLØ 71.
C USØ DE SUBPRØGRAMA SUBRØUTINE
      DIMENSIØN A(10),B(10)
      N = 10
      CALL LEA (A,N)
      CALL CALC(A,B,N,P)
      CALL IMP (A,B,N,P)
      STØP
      END
```

(Continúa)

(Continuación)

```
      SUBRØUTINE LEA (X,M)
      DIMENSIØN X(M)
      READ (1,51) X
      RETURN
   51 FØRMAT(10F3.1)
      END
      SUBRØUTINE CALC(X,Y,M,PR)
      DIMENSIØN X(M),Y(M)
      PR = PRØM (X,M)
      CALL ØRDEN(X,M)
      DØ 10 I=1,M
   10 Y(I) = X(I)*X(I)
      RETURN
      END
      FUNCTIØN PRØM (X,M)
      DIMENSIØN X(M)
      PRØM = 0.
      DØ 10 I=1,M
   10 PRØM = PRØM + X(I)
      PRØM = PRØM/M
      RETURN
      END
      SUBRØUTINE ØRDEN(Z,N)
      DIMENSIØN Z(N)
      M = N - 1
      DØ 15 I=1,M
      K = I + 1
      DØ 10 J=K,N
      IF(Z(I).LE.Z(J))GØ TØ 10
      AUX = Z(I)
      Z(I)= Z(J)
      Z(J) = AUX
   10 CØNTINUE
   15 CØNTINUE
      RETURN
      END
```

```
      SUBRØUTINE IMP (X,Y,M,PR)
      DIMENSIØN X(M),Y(M)
      WRITE(3,52)
      WRITE(3,53) X
      WRITE(3,54)
      WRITE(3,53) Y
      WRITE(3,55) PR
      RETURN
   52 FØRMAT(' MATRIZ DATØ'//)
   53 FØRMAT(' ',5F5.1)
   54 FØRMAT(' MATRIZ RESULTADØ'//)
   55 FØRMAT(' PRØMEDIØ',F7.2)
      END
```

El programa y subprogramas anteriores procesan los datos siguientes:

```
1  3  4  2  7 10  5  6  9  8
```

y entregan los resultados que aparecen a continuación:

```
MATRIZ DATØ

  1.0  2.0  3.0  4.0  5.0
  6.0  7.0  8.0  9.0 10.0
MATRIZ RESULTADØ

  1.0  4.0  9.0 16.0 25.0
 36.0 49.0 64.0 81.0100.0
PRØMEDIØ   5.50
```

c) <u>Proposición RETURN en subprogramas SUBROUTINE</u>

El retorno normal desde un subprograma FUNCTION o SUBROUTINE al programa llamador se efectúa con la proposición RETURN, sin argumento. En el primer caso, el retorno se realiza a la misma proposición que realizó el llamado, en el segundo, la vuelta es a la proposición siguiente a la proposición CALL que ejecutó el llamado.

En los subprogramas SUBROUTINE se puede volver a otros puntos del programa llamador, distintos del normal, mediante la estructura siguiente de la proposición RETURN.

i) Estructura de la proposición

RETURN i

donde:

i es una variable, sin subíndices, sin signo, que señala el punto de retorno.

ii) Función. Produce el retorno al programa llamador, al punto señalado por i. Entre los parámetros formales debe haber uno o más asteriscos separados entre sí, o de los otros parámetros formales, por coma. El valor de i está dado por:

1 ≤ valor de i ≤ número máximo de asteriscos

Cada uno de los asteriscos es reemplazado por un parámetro actual de la forma

εn ó $n

donde:

n: es un número de identificación de proposición

ε: se coloca si se utiliza el código EBCDIC

$: se coloca si se utiliza el código BCD

Si el valor de i es uno, el retorno se produce a la proposición cuyo número de identificación reemplazó al primer asterisco, si el valor de i es dos, a aquélla cuyo número reemplazó al segundo asterisco, etc.

Ejemplo 72:

```
C EJEMPLØ 72.
C RETØRNØ A PUNTØS DISTINTØS DEL NØRMAL.
      L = 0
    1 READ (1,50) I
      CALL SALTØ (I,$10,L,$20)
      WRITE(3,51) L
      IF(L.EQ.1) GØ TØ 1
      STØP
```

(Continúa)

(Continuación)

```
10 WRITE(3,52) L
   GØ TØ 1
20 WRITE(3,53) L
   STØP
50 FØRMAT(I2)
51 FØRMAT(' ',I3,' I < 0')
52 FØRMAT(' ',I3,' I = 0')
53 FØRMAT(' ',I3,' I > 0')
   END
   SUBRØUTINE SALTO (M,*,N,*)
   N = N + 1
   IF (M) 5,6,7
 5 RETURN
 6 RETURN 1
 7 RETURN 2
   END
```

El problema anterior lee los datos siguientes:

```
-5
 0
13
```

y entrega los resultados que aparecen a continuación:

```
ØØ1ØIØ<Ø0
ØØ2ØIØ=Ø0
ØØ3ØIØ>Ø0
```

E. Parámetros utilizados en subprogramas

a) Parámetros actuales

Parámetros actuales o reales son aquéllos que se transfieren al subprograma por el programa que lo llama. Reemplazan a los parámetros formales y deben corresponder en orden, número y tipo con éstos.

Los argumentos actuales pueden ser:

1) Cualquier tipo de constante, excepto la constante hexadecimal. Si se trata de una constante literal, debe especificarse de igual manera que en la proposición FORMAT, esto es, entre apóstrofos o precedida por wH; sin embargo, el valor pasado al subprograma corresponde sólo a la cadena de caracteres.

2) Cualquier tipo de variable, excepto las definidas por proposiciones ASSIGN

3) Cualquier tipo de nombres de arreglos (véase "Parámetros formales")

4) Cualquier tipo de expresión

5) Nombres de subprogramas (véase "Proposición EXTERNAL")

6) Números de identificación de proposiciones (sólo para subprogramas SUBROUTINE).

Ejemplo 73:

```
C EJEMPLØ 73.
C USØ DE LITERAL CØMØ PARAMETRØ ACTUAL
      IX = MAT('PRUEBAS')
      WRITE(3,52) IX
   52 FØRMAT(' ',I4)
      STØP
      END
      FUNCTIØN MAT(B)
      REAL*8 Y,B
      Y = B
      WRITE(3,52) Y
   52 FORMAT(' ',A12)
      MAT = 1
      RETURN
      END
```

El programa anterior entrega los resultados siguientes:

```
ØØØØPRUEBASØ
ØØØ1
```

b) Parámetros formales

Los parámetros formales se conocen también como parámetros vacíos, flotantes o fantasmas. Son reemplazados por los parámetros actuales y deben corresponder en orden, número y tipo con éstos.

En el caso de nombres de arreglos que figuren como parámetros formales, el parámetro actual debe ser:

i) para arreglos unidimensionales, un arreglo del mismo tipo y de dimensión igual o mayor que el del subprograma.

ii) para arreglos de más de una dimensión, un arreglo del mismo tipo y de dimensiones iguales. Se exceptúan los arreglos que se dimensionan en tiempo objeto.

Ninguno de los parámetros formales puede aparecer en proposiciones COMMON, EQUIVALENCE o NAMELIST.

Para un parámetro formal al cual se le asigna un valor dentro del subprograma, se debe hacer corresponder un parámetro actual que sea variable o arreglo, esto es, no debe utilizarse como parámetro actual una constante o expresión. El ejemplo que figura a continuación muestra lo que puede ocurrir al no cumplir con esta norma.

Ejemplo 74:

El programa que figura a continuación:

```
C EJEMPLØ 74.
C USØ DE SUBPRØGRAMA SUBRØUTINE
      INTEGER X,Y,Z
      READ (1,51) X,Y,Z
      CALL CAMBIØ(5,6,7)
      X = X * 5
      Y = Y * 6
      Z = Z * 7
      WRITE(3,52) X,Y,Z
      STØP
   51 FØRMAT(3I1)
   52 FØRMAT(' ',3I4)
      END
```

(Continúa)

(Continuación)

```
SUBRØUTINE CAMBIØ(I,J,K)
I = I + 1
J = J + 2
K = K + 3
RETURN
END
```

lee los datos siguientes:

567

Se definen así con la lectura las variables: X=5., Y=6. y Z=7. Dado que las variables X,Y y Z no han sido transferidas a la subrutina y, por lo tanto, no han sufrido cambio, los resultados que se deberían obtener serían:

| | | | |
|---|---|---|---|
| | X = X*5 | o sea | X = 25 |
| | Y = Y*6 | o sea | Y = 36 |
| y | Z = Z*7 | o sea | Z = 49 |

Sin embargo, al ser transferidas las <u>direcciones</u> de las constantes 5,6 y 7, reemplazaron a las <u>direcciones</u> de I, J y K respectivamente y sus contenidos fueron modificados, esto es, donde debería haber 5 quedó 6, en 6 quedó 8 y en 7 quedó 10. De ahí que el resultado que se obtiene es:

| | | | |
|---|---|---|---|
| | X = X*6 | o sea | X = 30 |
| | Y = Y*8 | o sea | Y = 48 |
| y | Z = Z*10 | o sea | Z = 70 |

ØØ30ØØ48ØØ70

F. <u>Llamadas de subprogramas</u>

El tipo de llamada de un subprograma está definido por la forma de especificar los parámetros formales. Se tiene así:

a) <u>Llamada por valor</u>

Corresponde a los subprogramas que aparecen en los ejemplos vistos. En ellos los parámetros formales aparecen separados entre sí por coma y todos encerrados entre paréntesis a continuación del nombre del subprograma. En este caso se reserva almacenamiento en el subprograma para los parámetros formales. Cuando el

subprograma es llamado, el valor del parámetro actual es llevado al almacenamiento reservado en aquél, desde el programa llamador. Cuando termina el proceso del subprograma, el resultado es transferido otra vez al argumento actual en el programa llamador.

b) Llamada por nombre

Los parámetros formales deben encerrarse entre operadores de división y separarse entre sí por coma. En este caso el subprograma no reserva almacenamiento para el parámetro formal. Para realizar los cálculos se utiliza el almacenamiento que corresponde al parámetro actual en el programa llamador.

Ejemplo 75:

```
C EJEMPLØ 75.
C LLAMADA PØR NØMBRE
      DIMENSIØN A(5),B(5),C(5)
      READ (1,51) A,B,N
      CALL XNØM(A,B,C,N,SUM)
      WRITE(3,52) SUM,A,B,C
      STØP
   51 FØRMAT(10F2.0,I2)
   52 FØRMAT(' ',F5.1/(' ',5F5.1))
      END
      SUBRØUTINE XNOM(/X/,/Y/,/Z/,/N/,/S/)
      DIMENSIØN X(N),Y(N),Z(N)
      S = 0.
      DØ 15 I=1,N
      S = S + X(I)*Y(I)
   15 Z(I) = X(I) + Y(I)
      RETURN
      END
```

El programa anterior lee los siguientes datos:

```
1 2 3 4 5 6 7 8 910 5
```

y entrega los resultados que figuran a continuación:

```
130.0
  1.0  2.0  3.0  4.0  5.0
  6.0  7.0  8.0  9.0 10.0
  7.0  9.0 11.0 13.0 15.0
```

G. Entradas múltiples en subprogramas

Aparte de la entrada normal a un subprograma SUBROUTINE, efectuada mediante la proposición CALL, y a un subprograma FUNCTION, realizada con la llamada del subprograma en una proposición de asignación, es posible tener múltiples entradas utilizando la proposición ENTRY dentro del subprograma llamado. Cada ENTRY en el subprograma define un punto de entrada distinto del normal. La proposición CALL o la referencia al subprograma FUNCTION utiliza esos puntos así definidos como nombre del subprograma.

a) Proposición ENTRY

La proposición ENTRY es no ejecutable y no afecta la secuencia durante la ejecución de un subprograma.

i) Estructura de la proposición

ENTRY nombre $(a_1, a_2, \ldots, a_n)$

donde:

nombre identifica un punto de entrada

los $a_i$ son parámetros formales

ii) Función. Se proporciona un punto de entrada distinto del normal en un subprograma. Los parámetros formales de la proposición ENTRY no necesitan corresponder en orden, tipo ni número con los parámetros formales del subprograma en el cual aparece, pero sí deben corresponder con los parámetros actuales de la proposición CALL o de la llamada del subprograma FUNCTION.

Los parámetros que se reciben por valor en alguna llamada previa del subprograma no necesitan aparecer como parámetros formales de la proposición ENTRY. En cambio, los que se reciben por nombre deben especificarse. En los subprogramas FUNCTION las proposiciones ENTRY deben tener al menos un parámetro formal.

Cuando hay una proposición ENTRY en un subprograma FUNCTION se debe asignar un valor, ya sea al nombre de la función o al nombre de entrada. Los tipos de estos nombres pueden ser diferentes; sin embargo, los nombres son considerados como equivalentes. Después que se asigna un valor a uno de ellos, los otros quedan automáticamente indefinidos en cuanto a valor.

El resultado devuelto por un subprograma FUNCTION es el último asignado al nombre de la función o a un nombre de entrada, independiente de que éste sea, o no, el nombre que aparece en la llamada. Si se ha asignado el valor a un nombre distinto y además éste difiere en tipo del nombre utilizado en la llamada, el valor de la función es indefinido.

Ejemplo 76:

```
C EJEMPLØ 76
C USØ DE PRØPØSICIØN ENTRY
      READ(1,50)ED,SD,PØRC
      ISW = -1
      DØ 10 I=1,3
      ISW = ISW + 1
      CALL RENTØ(ED,SD,PØRC,ISW,$1,$2)
      WRITE(3,51)
      GØ TØ 10
    1 WRITE(3,52)
      GØ TØ 10
    2 WRITE(3,53)
   10 CØNTINUE
      READ(1,50) PØRC
      ISW = -1
      DØ 15 I=1,3
      ISW = ISW + 1
      CALL RIML(PØRC,$3,$4,ISW)
      WRITE(3,51)
      GØ TØ 15
    3 WRITE(3,52)
      GØ TØ 15
```

(Continúa)

(Continuación)

```
 4 WRITE(3,53)
15 CØNTINUE
   ISW = -1
   DØ 20 I=1,3
   ISW = ISW + 1
   CALL RTAL($5,$6,ISW)
   WRITE(3,51)
   GØ TØ 20
 5 WRITE(3,52)
   GØ TØ 20
 6 WRITE(3,53)
20 CØNTINUE
   STØP
50 FØRMAT(3F4.0)
51 FØRMAT(' RETØRNØ NØRMAL ISW=1")
52 FØRMAT(' RETØRNØ N  1   ISW=0')
53 FØRMAT(' RETØRNØ N  2   ISW=2')
   END
   SUBRØUTINE RENTØ(VD,RB,/PI/,ISW,*,*)
   RBRU = VD * RB
   ENTRY RIML(/PI/,*,*,ISW)
   RLEG = RBRU * PI
   ENTRY RTAL(*,*,ISW)
   RLIQ = RBRU - RLEG
   IF(ISW - 1) 10,11,12
10 WRITE(3,51) RBRU
   RETURN 1
11 WRITE(3,52) RBRU,RLEG
   RETURN
12 WRITE(3,53) RBRU,RLEG,RLIQ
   RETURN 2
51 FØRMAT(' RB=',F14.2/)
52 FØRMAT(' RB=',F14.2/' RI=',F14.2/)
53 FØRMAT(' RB=',F14.2/' RI=',F14.2/' RL=',F14.2/)
   END
```

El programa y subprograma anteriores procesan los datos siguientes:

```
3500 6500.1
0.15
```

y entregan los resultados que figuran a continuación:

```
RB=     2275000.00

RETØRNØ N  1   ISW=0
RB=     2275000.00
RI=      227499.88

RETØRNØ NØRMAL ISW=1
RB=     2275000.00
RI=      227499.88
RL=     2047500.00

RETØRNØ N  2   ISW=2
RB=     2275000.00

RETØRNØ N  1   ISW=0
RB=     2275000.00
RI=      341249.94

RETØRNØ NØRMAL ISW=1
RB=     2275000.00
RI=      341249.94
RL=     1933750.00

RETØRNØ N  2   ISW=2
RB=     2275000.00

RETØRNØ N  1   ISW=0
RB=     2275000.00
RI=      341249.94

RETØRNØ NØRMAL ISW=1
RB=     2275000.00
RI=      341249.94
RL=     1933750.00

RETØRNØ N  2   ISW=2
```

H. Proposición EXTERNAL

i) Estructura de la proposición

EXTERNAL $a_1, a_2, a_3, \ldots, a_n$

donde:

los $a_i$ son nombres de subprogramas que se pasan como parámetros actuales a otro(s) subprograma(s)

ii) Función. Es una proposición de especificación que declara como símbolos externos al programa llamador aquellos nombres de subprogramas que se pasan como parámetros actuales a otro u otros subprogramas.

Debe preceder las definiciones de funciones de proposición y a todas las proposiciones ejecutables.

Ejemplo 77:

Se tienen los siguientes datos:

5 1 3 5 7 9 2 4 6 810

que son procesados por el programa y subprogramas que figuran a continuación:

Ejemplo 77:

```
C EJEMPLØ 77.
C USØ DE PRØPØSICIØN EXTERNAL
      EXTERNAL AMUL,REST
      DIMENSIØN X(5),Y(5),Z(5)
      READ (1,51) M,X,Y
      N = -1
      CALL SUB(X,Y,Z,N,M,W,AMUL)
      WRITE(3,52)W,X,Y,Z
      CALL SUB(X,Y,Z,N,M,W,REST)
      WRITE(3,52)W,Z
      N = 2
      CALL SUB(X,Y,Z,N,M,W,AMUL)
      WRITE(3,52)W,Z
      CALL SUB(X,Y,Z,N,M,W,REST)
      WRITE(3,52)W,Z
      STØP
```

(Continúa)

(Continuación)

```
51 FØRMAT(I2,10F2.0)
52 FØRMAT(' ',F7.2/(5F7.2))
   END
   SUBRØUTINE SUB(A,B,C,/N/,/M/,/X/,RUT)
   DIMENSIØN A(M),B(M),C(M)
   IF(N)1,1,7
 1 X = RUT(A,B,C,M)
 2 IF(X)3,6,4
 3 X = -X
 4 X = SQRT(X)
 5 RETURN
 6 X = 0.
   RETURN
 7 X = RUT(B,A,C,M)
   GØ TØ 2
   END
   FUNCTIØN AMUL(A,B,C,/M/)
   DIMENSIØN A(M),B(M),C(M)
   AMUL = 0.
   DØ 15 I=1,M
   IF(B(I))5,10,5
 5 C(I) = A(I)/B(I)
   GØ TØ 15
10 C(I) = 0.
15 AMUL = AMUL + C(I)
   RETURN
   END
   FUNCTIØN REST(A,B,C,/M/)
   DIMENSIØN A(M),B(M),C(M)
   REST = 0
   DØ 15 I=1,M
   C(I) = A(I) - B(I)
15 REST = REST + C(I)
   RETURN
   END
```

Los resultados entregados son:

```
ƀƀƀ1.96
ƀƀ1.00ƀƀƀ3.00ƀƀƀ5.00ƀƀƀ7.00ƀƀƀ9.00
ƀƀ2.00ƀƀƀ4.00ƀƀƀ6.00ƀƀƀ8.00ƀƀ10.00
ƀƀ0.50ƀƀƀ0.75ƀƀƀ0.83ƀƀƀ0.88ƀƀƀ0.90
ƀƀƀ2.24
ƀ-1.00ƀƀ-1.00ƀƀ-1.00ƀƀ-1.00ƀƀ-1.00
ƀƀƀ2,61
ƀƀ2.00ƀƀƀ1.33ƀƀƀ1.20ƀƀƀ1.14ƀƀƀ1.11
ƀƀƀ2.24
ƀƀ1.00ƀƀƀ1.00ƀƀƀ1.00ƀƀƀ1.00ƀƀƀ1.00
```

## I. Proposiciones de especificación y subprogramas.

### a) Proposición COMMON

i) Estructura de la proposición

COMMON $/r_1/a_{11}(k_{11}),a_{12}(k_{12}),\ldots/r_n/a_{n1}(k_{n1}),a_{n2}(k_{n2}),\ldots$

donde:

los $a_{ij}$ son nombres de variables o arreglos que no sean parámetros formales en subprogramas (véase "Subprogramas")

los $k_{ij}$ son opcionales y representan las dimensiones del arreglo (véase "Proposición DIMENSION")

los $r_i$ son opcionales y representan nombres de bloque común. Deben ser encerrados entre operadores de división. Si no se especifican caracteres entre los operadores o se especifican blancos, se define un área común en blanco. Si $r_1$ denota un área común en blanco, los operadores pueden eliminarse.

ii) Función. Define un área de memoria que será compartida por dos o más módulos de programa. Especifica, además, las variables o arreglos que ocuparán esta área. Dado que el almacenamiento compartido, en la práctica corresponde a zonas de memoria que están identificadas por dos o más nombres, cada uno de ellos en el respectivo módulo, debe haber correspondencia de orden, longitud y tipo entre esos nombres.

En el caso de arreglos, es posible especificar las dimensiones y éstas deben ser constantes enteras sin signo. No pueden aparecer en una proposición COMMON arreglos que serán dimensionados en tiempo de ejecución.

Si aparece más de una proposición COMMON en un módulo, son equivalentes a una sola en la que se acumulan todas las variables y arreglos en el mismo orden de aparición. No puede haber nombres de variables o arreglos repetidos.

Si no se especifica un nombre entre operadores de división, el área común creada se denomina "Area común en blanco". Es posible definir áreas comunes separadas y éstas se identifican con un nombre especificado entre operadores de división. Esto permite a un módulo compartir un área común con un segundo módulo y otra área común con un tercero. Dichas áreas se denominan "Areas comunes con nombre".

El área común en blanco se define automáticamente al no aparecer nombre ni operadores de división al comienzo de la proposición o al colocarse dos operadores de división seguidos, con o sin blanco entre ellos, al comienzo o después de él, en la proposición COMMON. La reaparición de operadores de división con o sin blanco entre ellos, o con un nombre que ha sido definido anteriormente, representa la continuación del área respectiva.

El orden en que deben especificarse las variables en la proposición COMMON debe ser el siguiente:

1° variables de 8 bytes
2° variables de 4 bytes
3° variables de 2 bytes
4° variables de 1 byte

Ejemplo 78:

```
COMMON    A,B,C
COMMON    D,X(10)
```

es equivalente a:

```
COMMON    A,B,C,D,X(10)
```

o también a:

```
DIMENSION X(10)
COMMON    A,B,C,D,X
```

Ejemplo 79:

```
COMMON  I,J/ALFA/X,Y,Z//K,L/ALFA/U,V,W
```

define un área común en blanco que contiene las variables I,J,K y L y un área común de nombre ALFA que contiene las variables X,Y,Z,U,V y W.

Ejemplo 80:

Si se tienen dos variables, A y B, y un arreglo C de cinco por cinco elementos, todos de doble precisión; dos variables, D y E, de precisión simple; tres variables enteras, I,J,K, de cuatro bytes de longitud; dos variables enteras, M y N, de dos bytes de longitud y una variable lógica L, de un byte de longitud, las especificaciones y orden correctos deben ser:

```
REAL*8    A,B,C(5,5)
INTEGER*2 M,N
LOGICAL*1 L
COMMON A,B,C(5,5),D,E,I,J,K,M,N,L
```

Si no se conserva el orden indicado en cuanto a longitudes, deben especificarse variables artificiales que permitan lograr el alineamiento que corresponde a cada variable.

Ejemplo 81:

Si las variables del ejemplo 80 se especifican en el orden siguiente: D,A,B,I,C(5,5),M,J,K,L,N,E, las proposiciones tendrán que ser:

```
INTEGER*2   AUX3
COMMON  D,AUX1,A,B,I,AUX2,C(5,5),M,AUX3,J,K,L
COMMON  AUX4,N,E
LOGICAL*1  AUX4
```

La primera variable del COMMON parte siempre con el alineamiento de ocho bytes. En este caso, D queda en esas condiciones, pero dado que tienecuatro bytes, es necesario AUX1 para que queden con alineamiento correcto A y B. En igual forma, como I tiene cuatro bytes, se necesita AUX2 para dar alineamiento correcto a C(5,5). A continuación, como M ocupa dos bytes, se necesita AUX3 (longitud dos bytes) para que queden bien ubicados J y K. Finalmente, como L tiene un byte de longitud, se necesita AUX4 de igual largo para que la variable N quede en forma correcta. Al sacar la cuenta de los bytes ocupados, se observará que la variable E está con el alineamiento adecuado.

Ejemplo 82:

```
C EJEMPLØ 82.
C USØ DE PRØPØSICIØN CØMMØN
      DIMENSIØN Ø(5),P(5)
      COMMØN A,B/G1/C(5),D(5)//E,M/G1/G
      READ (1,50)N,(C(I),D(I),I=1,N),M,Ø,A,B,E
      L = 1
      G = 0.
      CALL SUB1(N,L)
      WRITE(3,51)C,D,G,N,L
      CALL SUB2(Ø,P)
      WRITE(3,52)A,B,E,M,Ø,P
      STØP
   50 FØRMAT(I2,10F2.0/I2,5F2.0,3F3.1)
   51 FØRMAT(' ',10F4.0,F6.1,2I3/)
   52 FØRMAT(' ',3F5.1,I4/' ',10F5.1)
      END
      SUBRØUTINE SUB1(N,L)
      DIMENSIØN U(5),V(5)
      CØMMØN /G1/U,V,C
      DØ 10 I=L,N
   10 C = C + U(I)*V(I)
      RETURN
      END

      SUBRØUTINE SUB2 (/X/,/Z/)
      DIMENSIØN X(N),Z(N)
      CØMMØN A,B,C,N
      DØ 10 I=1,N
   10 Z(I) = (X(I)+A*B)/C
      RETURN
      END
```

El programa y los subprogramas anteriores procesan los datos siguientes:

```
5 1 2 3 4 5 6 7 8 910
515202530352.54.0100
```

y entregan los resultados que figuran a continuación:

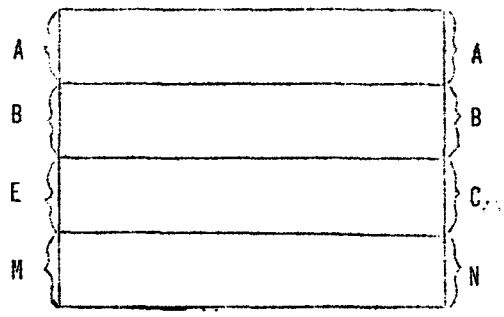
```
ØØ1.ØØ3.ØØ5.ØØ7.ØØ9.ØØ2.ØØ4.ØØ6.ØØ8.Ø10.Ø190.0ØØ5ØØ1

ØØ2.5ØØ4.0Ø10.0ØØØ5
Ø15.0Ø20.0Ø25.0Ø30.0Ø35.0ØØ2.5ØØ3.0ØØ3.5ØØ4.0ØØ4.5
```

En el ejemplo anterior la proposición COMMON del programa principal crea un área común en blanco que contiene las variables A,B,E y M, y un área común de nombre G1 que contiene los arreglos C(5),D(5) y la variable G.

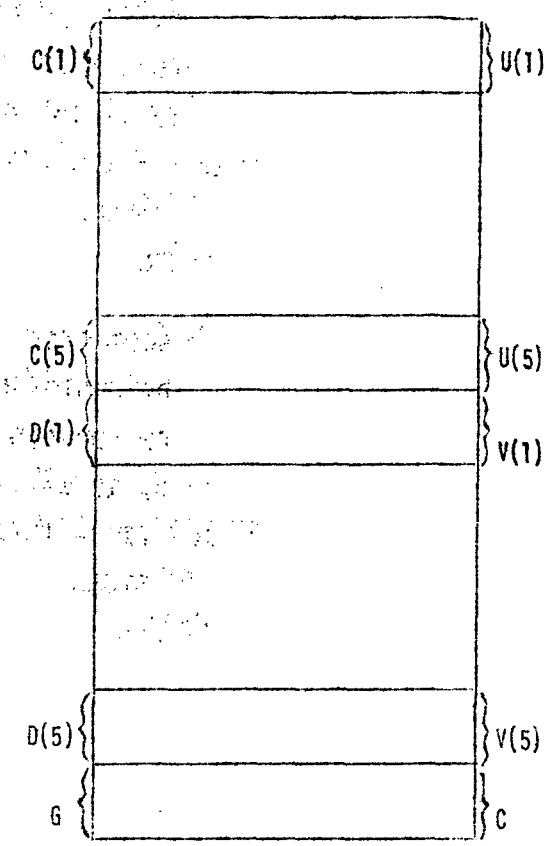
El área común en blanco es compartida con el subprograma SUB2 y el área común G1 con el subprograma SUB1.

El esquema de distribución de dichas áreas, que aparece a continuación, muestra las áreas compartidas y sus respectivos nombres.

Area común en blanco




Area común G1



b) Proposición EQUIVALENCE

i) Estructura de la proposición

EQUIVALENCE($a_{11}$,$a_{12}$,$a_{13}$,...),($a_{21}$,$a_{22}$,$a_{23}$,...),...

donde:

los $a_{ij}$ pueden ser variables con o sin subíndices. Estos se pueden expresar considerando el arreglo como unidimensional en cuyo caso la posición indicada es relativa al primer elemento o considerando las dimensiones reales, esto es, en cada dimensión la posición es relativa al primer elemento de dicha dimensión. En ambos casos los subíndices deben ser constantes enteras sin signo.

ii) Función. Se declaran, encerradas entre paréntesis, todas las variables que comparten memoria dentro del módulo de programa. Todas las variables deben ser del mismo tipo y longitud.

La equivalencia entre dos elementos de arreglos distintos implica la equivalencia de otros elementos de esos mismos arreglos, a causa del orden de almacenamiento preestablecido para ellos.

No pueden hacerse equivalentes variables que estén dentro de un área común o que pertenezcan a diferentes áreas comunes.

Una variable que esté en un área común se puede hacer equivalente a una variable que no lo esté. Si ésta última es un elementos de arreglo, se puede conseguir aumentar el tamaño del área común, que es válido cuando el límite superior del área se desplaza hacia adelante y no cuando el límite inferior o comienzo del área común se desplaza hacia atrás.

Ejemplo 83:

```
DIMENSION   B(5),C(10,10),D(5,10,15)
EQUIVALENCE (A,B(1),C(5,3)),(D(5,10,2),E)
```

La proposición EQUIVALENCE indica que las variables A,B(1) y C(5,3) comparten la misma posición de memoria. A partir de las variables B(1) y C(5,3), el resto de los elementos de los arreglos B y C comparten memoria de acuerdo con su ubicación en el respectivo arreglo. La proposición especifica también que el elemento D(5,10,2) comparte memoria con la variable E.

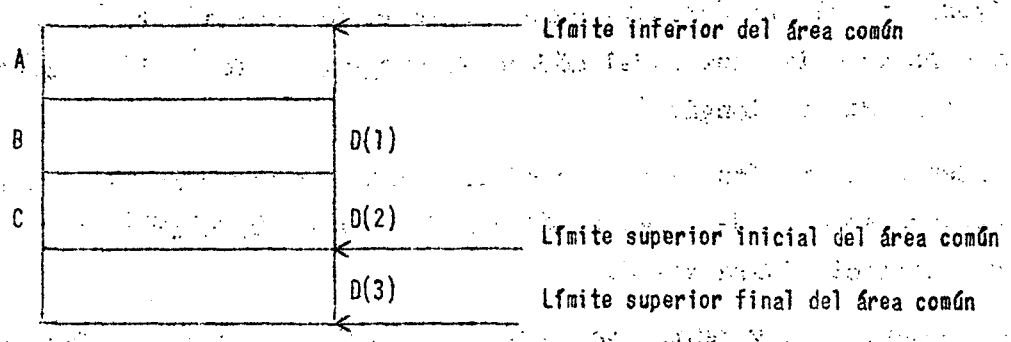
Se obtiene el mismo efecto al especificar

```
EQUIVALENCE   (A,B(1),C(25)),(D(100),E)
```

Ejemplo 84:

```
COMMON      A,B,C
DIMENSION   D(3)
EQUIVALENCE (B,D(1))
```

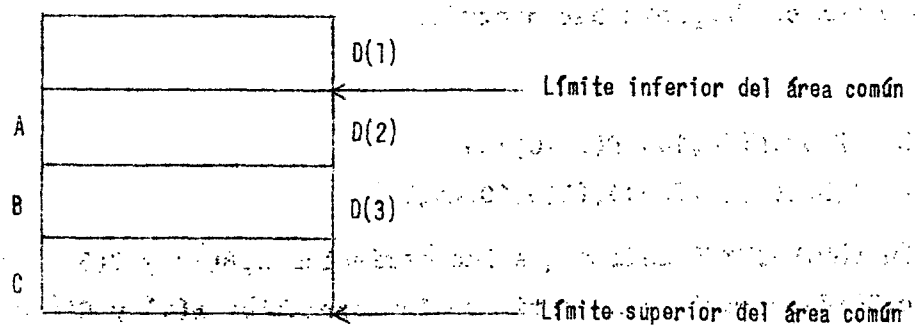
La proposición COMMON genera un área común para las variables A,B y C. La proposición EQUIVALENCE causará que la variable D(1)comparta memoria con B, D(2) con C y D(3) extiende el límite del área común como se indica a continuación:




Si se especifica

```
EQUIVALENCE  (B,D(3))
```

se produciría un error debido a que se fuerza a D(1) a ocupar posiciones anteriores al límite inferior del área común.



```
C EJEMPLØ 85.
C USØ DE PRØPØSICIØN EQUIVALENCE
      DIMENSIØN A(9)
      COMMØN B(3,3)
      EQUIVALENCE (A(6),B(9))
      READ (1,51) B,A
      CALL DIAG(3,3)
      WRITE(3,52)
      WRITE(3,53)
      STØP
   51 FØRMAT(9F2.0)
   52 FØRMAT(' PRØBLEMA DE PRUEBA')
   53 FØRMAT(T10,'MATRIZ RESULTADØ'////9F4.1)
      END
      SUBRØUTINE DIAG(M,N)
      CØMMØN X(3,3)
      DØ 10 I = 1,M
      DØ 10 J = 1,N
      IF(J-I) 20,30,20
   30 X(I,J) = 0.
      GØ TØ 10
   20 X(I,J) = X(J,I)
   10 CØNTINUE
      RETURN
      END
```

El programa y subprograma anteriores procesan los datos siguientes:

```
1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1
```

y entregan los resultados que figuran a continuación:

```
PRØBLEMA DE PRUEBA
ØØØØØØØØØMATRIZ RESULTADØ

0.0Ø9.0Ø6.0Ø9.0Ø0.0Ø5.0Ø6.0Ø5.0Ø0.0
```

J. Dimensionamiento en tiempo de ejecución.

Las dimensiones absolutas de arreglos utilizados en subprogramas no necesitan especificarse mediante constantes enteras. Pueden indicarse en proposiciones DIMENSION o en proposiciones de especificación de tipo explícitas, mediante variables enteras de cuatro bytes de longitud.

En el momento de ser llamado el subprograma, las variables enteras que representan dimensiones son reemplazadas por los parámetros reales transferidos al subprograma. La transferencia se realiza a través de parámetros formales o mediante la proposición COMMON.

Deben considerarse cuatro elementos en total para efectuar un dimensionamiento correcto. Estos elementos son:

conjunto actual en el programa llamador
dimensiones absolutas del conjunto actual
conjunto formal en el subprograma
dimensiones actuales transferidas al subprograma.

El nombre del conjunto actual reemplazará al nombre del conjunto formal en el subprograma. Las dimensiones actuales transferidas deben ser iguales a las dimensiones absolutas del conjunto actual, se exceptúa el caso de arreglos unidimensionales para los cuales pueden ser inferiores o iguales.

El nombre de un arreglo con dimensiones de tiempo de ejecución no puede figurar en una proposición COMMON.

El tamaño de la dimensión variable puede ser transferido a través de más de un nivel de subprogramas.

K. Problemas propuestos

a) Escribir una función de proposición para calcular

i) raíz cúbica de x
ii) sen(2x) = 2sen x cos x
iii) $D = ((x_1-x_2)^2 + (y_1-y_2)^2)^{1/2}$

b) Escribir un subprograma función que calcule la raíz máxima de una ecuación de segundo grado.

c) Hacer una subrutina que calcule el producto de dos matrices A y B de acuerdo con la fórmula general:

C(L,N) = A(L,M)*B(M,N)

El nombre de la subrutina y sus parámetros formales deben ser:

PMAT(A,B,C,L,M,N)

d) Hacer una subrutina que determine el triángulo de área mayor, dada una lista que contiene los lados de N triángulos consecutivos.

e) Hacer un subprograma que calcule los ángulos internos de un triángulo, por el teorema general de Pitágoras ($c^2 = a^2+b^2-2ab(a,b)$). Se dan las coordenadas de los tres vértices. Los valores de los ángulos deben ser devueltos en grados sexagesimales al programa principal.

f) Hacer un subprograma que ajuste una recta: y=a+bx, a una serie de puntos

$$a = \frac{\sum x_i^2 \sum y_i - \sum x_i y_i \sum x_i}{N \sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{N \sum y_i x_i - \sum y_i \sum x_i}{N \sum x_i^2 - (\sum x_i)^2}$$

g) Hacer una subrutina que integre una función por el método de los trapecios, entre los límites a y b. El valor de f(x) debe ser calculado con una función de proposición en el programa principal.

$$\text{AREA} = \int_a^b f(x)dx = \frac{b'}{2}(y_o+2y_1+2y_2+\ldots+2y_{n-1}+y_n)$$

donde $b' = \frac{b-a}{n}$

h) Hacer un subprograma para calcular:

$$e^x = 1 + \frac{x}{-\frac{x}{2} - \left(\frac{k_o + k_1 x^2}{1 + k_2 x^2}\right)}$$

para $-\log\sqrt{2} \leq x \leq \log\sqrt{2}$

$k_o = 1.0000000020967$
$k_1 = 0.0999743507186$
$k_2 = 0.0166411490538$

i) Escribir un programa que utilice un subprograma SBP1 para calcular

$$h = \sqrt{a^2 - x^2}\ \mathrm{Ln}(b) + \frac{|\mathrm{Ln}(c)|}{\sqrt{a^2 + x^2}} - \frac{1}{b^2}\mathrm{arctg}\ \frac{x}{a}$$

Todos los datos están perforados en una tarjeta

a ocupa las columans 1 a 6 (4 díg.enteros, 2 decimales)
b ocupa las columnas 7 a 12 (4 díg. enteros, 2 decimales)
c ocupa las columnas 21 a 25 (3 díg. enteros, 2 decimales)
x ocupa las columnas 26 a 31 (4 díg. enteros, 2 decimales)

Los resultados deben salir impresos en la siguiente forma:

Primera línea posición 15 RESULTADOS DEL PROGRAMA
Quinta línea posición 10 H = posición 15 el valor de H con F6.3

j) Una tarjeta perforada contiene lo siguiente:

ØXØ=Ø0.3723ØØØIAØ=ØØ3ØØØYØ=Ø0.3117E+01
A KA B

Programar una subrutina que lea una serie de N tarjetas de este tipo y deje los datos en las listas A,KA y B respectivamente. El nombre de la subrutina y sus parâmetros formales deben ser:

LECT(A,KA,B,N)

Utilizar esta subrutina para leer N grupos de datos de este tipo, para definir las listas X,IA,Y (N < 1000) y formar una nueva lista Z de la siguiente manera:

si $IA_i < 0$ entonces $Z_i = Y_i/X_i^2$

$IA_i = 0$ entonces $Z_i = Y_i + \sqrt{5X_i}$

$IA_i > 0$ entonces $Z_i = \sqrt{\cos(Xi/Y_i)}$

Imprimir pares de IA y Z

k) Programar la transformación de un arreglo bidimensional A(n,n) en un arreglo unidimensional B(m), donde $m=n^2$. Entregar el arreglo unidimensional B a un subprograma que verifica a base de este arreglo, si el original A es simêtrico. Si lo es, hacer SIM=0 e imprimir y si no lo es, hacer SIM=1 e imprimir dicho valor.

l) Hacer una subrutina ELEM (M,N) que elimine elementos repetidos de una lista que se pasa a través de una proposición COMMON.

M = largo de la lista original

N = largo de la lista resultado

m) Se tiene una lista de edades (1000 datos), 40 por tarjeta. Se pide ordenarlos de mayor a menor y formar dos archivos en cinta magnética de acuerdo con el criterio siguiente:

archivo 1 : menores o iguales a 40 años

archivo 2 : mayores de 40 años

El ordenamiento de los datos debe hacerse en un subprograma al cual se le pasarán los datos a través de una proposición COMMON.

n) Analizar el siguiente programa e indicar qué resultados entrega:

```
DIMENSIØN B(10)
CØMMØN A(10),TMC
EQUIVALENCE (A(5),B(5))
READ (1,51) A
WRITE(8) A
END FILE 8
CALL CUADR(10)
```

(Continúa)

(Continuación)

```
      DØ 10 J = 1,10
   10 B(J) = A(J) - TMC
      WRITE(3,52) A
      REWIND 8
      READ (8) B
      WRITE(3,52)(A(K),K=1,10)
      STØP
   51 FØRMAT(10F3.0)
   52 FØRMAT('0',10(F3.0,2X))
      END
      SUBRØUTINE CUADR(N)
      CØMMØN C(10),TMC
      REWIND 8
      READ (8) C
      SUM = 0.
      DØ 10 J = 1,N
   10 SUM = SUM + C(I)**2
      TMC = SQRT(SUM)
      RETURN
      END
```

Datos:

1. 4. 4. 5. 2. 1. 2. 1. 7. 2.

## 6. Proposiciones para depuración de programas

Se entiende por depuración de programas la "limpieza" que se hace de ellos para eliminarles todos los errores que se puedan producir durante el proceso y que son difíciles de localizar a simple vista por la complejidad del proceso, cantidad de subprogramas, variedad de datos procesados, etc.

Las proposiciones para depuración forman un paquete que debe ser colocado entre el programa de solución del problema y la proposición END.

A. Proposición DEBUG

Debe haber una proposición DEBUG por cada programa o subprograma que va a ser depurado y debe estar inmediatamente antes del paquete de depuración.

i) Estructura de la proposición

DEBUG opción 1, opción 2,...,opción n

donde:

opción i puede ser cualquiera de las siguientes:

1) UNIT (a)

donde:

a es una constante entera sin signo, que representa un número de referencia de conjunto de datos. Todos los resultados producidos por el paquete de depuración salen a través del dispositivo asimilado al número a.

2) SUBCHK ($a_1$,$a_2$,...,$a_n$)

donde:

los $a_i$ representan nombres de arreglos.

Al especificar esta opción se verifica la validez de los subíndices usados con los nombres de arreglos indicados. Para ello se utilizan como referencia las dimensiones declaradas para cada arreglo. Si se omite la lista de nombres y sólo se especifica SUBCHK se verifican los subíndices de todos los arreglos. Si algún subíndice se sale de rango, se emite un mensaje y el proceso continúa con el subíndice incorrecto.

3) TRACE

Si se especifica, se obtiene el trazado o recorrido que efectúa el proceso dentro de un programa. El trazado queda indicado por la especificación de los números de identificación de las proposiciones que se ejecutan, la cual se obtiene a través del dispositivo señalado en la opción UNIT.

Además de indicar esta opción, debe utilizarse conjuntamente la proposición TRACE ON en el paquete de depuración.

4) INIT $(n_1, n_2, \ldots, n_n)$

donde:

los $n_i$ representan nombres de variables o arreglos

Especificando esta opción, se emite el nombre de una variable o del elemento de arreglo, y el valor respectivo, cada vez que éste cambia. Si se omite la lista, se entrega el nombre de cualquier variable o elemento de arreglo que cambie de valor.

5) SUBTRACE

Al colocar esta opción en la depuración de un subprograma, se emite el nombre de éste, cada vez que es llamado, y al salir del subprograma se entrega el mensaje RETURN.

ii) Función. Permite especificar las operaciones de depuración que se ejecutarán en el programa principal o en los subprogramas.

B. Proposición AT

i) Estructura de la proposición

AT n

donde:

n es un número de identificación de proposición ejecutable.

ii) Función. Identifica el comienzo de un paquete de depuración e indica el punto en el programa donde se inciará dicha depuración.

Las operaciones de depuración especificadas en el paquete son realizadas antes de la ejecución de la proposición identificada con n.

C. Proposición TRACE ON

i) Estructura de la proposición

TRACE ON

ii) Función. Permite obtener el trazado o recorrido que efectúa el proceso, dentro de un programa. Debe especificarse, sin embargo, la opción TRACE en la proposición DEBUG.

D. Proposición TRACE OFF

i) Estructura de la proposición

TRACE OFF

ii) Función. Pone término al trazado iniciado con la proposición TRACE ON. El punto donde se termina el trazado se especifica con la proposición AT ubicada inmediatamente antes de la proposición TRACE OFF.

E. Proposición DISPLAY

i) Estructura de la proposición

DISPLAY lista

donde:

lista es un conjunto de nombres de variables y arreglos, separados entre sí por coma.

ii) Función. Se emite el nombre de la variable y su valor. Si se trata de arreglos, se entrega el nombre del arreglo y los valores de sus elementos.

El efecto producido por una proposición DISPLAY lista, es el mismo que se obtiene con:

```
NAMELIST  /nombre/lista
WRITE (n,nombre)
```

No se pueden emitir valores de variables o arreglos que figuren como parámetros formales.

Ejemplo 86:

```
C EJEMPLØ 86.
C USØ DE PRØPØSICIØNES DE DEPURACIØN
      DIMENSIØN A(10),B(10)
      N = 10
      CALL LEA (A,N)
      CALL CALC(A,B,N,P)
      CALL IMP (A,B,N,P)
      STØP
      END
```

(Continúa)

(Continuación)

```
      SUBRØUTINE LEA (X,M)
      DIMENSIØN X(M)
      READ (1,51) X
      RETURN
   51 FØRMAT(10F3.1)
      DEBUG UNIT(3),SUBTRACE
      END
      SUBRØUTINE CALC(X,Y,M,PR)
      DIMENSIØN X(M),Y(M)
    1 PR = PRØM (X,M)
      CALL ØRDEN(X,M)
      DØ 10 I=1,M
   10 Y(I) = X(I)*X(I)
    2 RETURN
      DEBUG UNIT(3),SUBTRACE
      AT 1
      TRACE ØN
      AT 2
      TRACE ØFF
      END
      FUNCTIØN PRØM (X,M)
      DIMENSIØN X(M)
   21 PRØM = 0.
      DØ 10 I=1,M
   10 PRØM = PRØM + X(I)
      PRØM = PRØM/M
   22 RETURN
      DEBUG UNIT(3),SUBTRACE,TRACE,INIT(I,PRØM)
      AT 21
      TRACE ØN
      AT 22
      TRACE ØFF
      END
```

```
      SUBRØUTINE ØRDEN(Z,N)
      DIMENSIØN Z(N)
   31 M = N - 1
      DØ 15 I=1,M
      K = I + 1
      DØ 10 J=K,N
      IF(Z(I).LE.Z(J)) GØ TØ 10
      AUX = Z(I)
      Z(I)= Z(J)
      Z(J)= AUX
   10 CØNTINUE
   15 CØNTINUE
   32 RETURN
      DEBUG UNIT(3),SUBTRACE,TRACE,INIT(Z)
      AT 31
      TRACE ØN
      DISPLAY AUX
      AT 32
      TRACE ØFF
      END
      SUBRØUTINE IMP (X,Y,M,PR)
      DIMENSIØN X(M),Y(M)
   41 WRITE (3,52)
      WRITE (3,53)X
      WRITE (3,54)
      WRITE (3,53)Y
      WRITE (3,55)PR
      RETURN
   52 FØRMAT(' MATRIZ DATØ'//)
   53 FØRMAT(' ',5F5.1)
   54 FØRMAT(' MATRIZ RESULTADØ'//)
   55 FØRMAT(' PRØMEDIØ',F7.2)
      DEBUG UNIT(3),SUBTRACE
      AT 41
      DIMENSIØN W2(10),W3(10)
      W4 = PR
```

(Continúa)

(Continuación)

```
      DØ 200 IW2=1,10
      W2(IW2) = X(IW2)
  200 W3(IW2) = Y(IW2)
      DISPLAY W2,W3,W4
      END
```

El programa y subprogramas anteriores leen los datos siguientes:

Ø7ØØ5ØØ3ØØ9ØØ4ØØ1ØØ8ØØ2ØØ6Ø10Ø

y entregan los resultados que figuran a continuación. Por razones de espacio, éstos se han agrupado en dos columnas sobre las cuales se especifica un número que indica el orden en que deben ser leídas.

1

```
SUBTRACE LEA
SUBTRACE *RETURN*
SUBTRACE CALC
SUBTRACE PRØM
TRACE 21
PRØM =  0.0
I = 1
TRACE 10
PRØM =   7.000000
I = 2
TRACE 10
PRØM =   12.00000
I = 3
TRACE 10
PRØM =   15.00000
I = 4
TRACE 10
PRØM =   24.00000
I = 5
TRACE 10
PRØM =   28.00000
I = 6
TRACE 10
```

2

```
PROM =   29.00000
I = 7
TRACE 10
PRØM =   37.00000
I = 8
TRACE 10
PRØM =   39.00000
I = 9
TRACE 10
PRØM =   45.00000
I = 10
TRACE 10
PRØM =   55.00000
PRØM =   5.500000
SUBTRACE *RETURN*
SUBTRACE ØRDEN
%DBGOO#
AUX=-0.17296600
%END
TRACE 31
Z(1) =   5.000000
Z(2) =   7.000000
TRACE 10
```

3

```
Z(1) =   3.000000
Z(3) =   5.000000
TRACE 10
TRACE 10
TRACE 10
Z(1) =   1.000000
Z(6) =   3.000000
TRACE 10
TRACE 10
TRACE 10
TRACE 10
TRACE 10
TRACE 15
Z(2) =   5.000000
Z(3) =   7.000000
TRACE 10
TRACE 10
Z(2) =   4.000000
Z(5) =   5.000000
TRACE 10
Z(2) =   3.000000
Z(6) =   4.000000
TRACE 10
TRACE 10
Z(2) =   2.000000
Z(8) =   3.000000
TRACE 10
TRACE 10
TRACE 10
TRACE 15
TRACE 10
Z(3) =   5.000000
Z(5) =   7.000000
```

4

```
TRACE 10
TRACE 10
TRACE 10
TRACE 15
Z(4) =   7.000000
Z(5) =   9.000000
TRACE 10
Z(4) =   5.000000
Z(6) =   7.000000
TRACE 10
TRACE 10
Z(4) =   4.000000
Z(8) =   5.000000
TRACE 10
TRACE 10
TRACE 10
TRACE 15
Z(5) =   7.000000
Z(6) =   9.000000
TRACE 10
TRACE 10
Z(5) =   5.000000
Z(8) =   7.000000
TRACE 10
TRACE 10
TRACE 10
TRACE 15
Z(6) =   8.000000
Z(7) =   9.000000
TRACE 10
Z(6) =   7.000000
Z(8) =   8.000000
TRACE 10
Z(6) =   6.000000
```

(Continúa)

(Continuación)

3

```
TRACE 10
Z(3) =   4.000000
Z(6) =   5.000000
TRACE 10
TRACE 10
Z(3) =   3.000000
Z(8) =   4.000000
```

4

```
Z(9) =   7.000000
TRACE 10
TRACE 10
TRACE 15
Z(7) =   8.000000
Z(8) =   9.000000
TRACE 10
```

5

```
Z(7) =   7.000000
Z(9) =   8.000000
TRACE 10
TRACE 10
TRACE 15
Z(8) =   8.000000
Z(9) =   9.000000
TRACE 10
TRACE 10
TRACE 15
TRACE 10
TRACE 15
SUBTRACE *RETURN*
SUBTRACE *RETURN*
SUBTRACE IMP
$DBG00#
W2=  1.0000000    ,  2.0000000    ,
  3.0000000    ,  4.0000000    ,
  5.0000000    ,  6.0000000    ,
  7.0000000    ,  8.0000000    ,
  9.0000000    ,  10.000000    ,W3=
  1.0000000    ,  4.0000000    ,
  9.0000000    ,  16.000000    ,
```

(Continúa)

(Continuación)

5

```
  25.000000    ,  36.000000    ,
  49.000000    ,  64.000000    ,
  81.000000    ,  100.00000    ,W4=
  5.5000000
$END
MATRIZ DATØ

  1.0  2.0  3.0  4.0  5.0
  6.0  7.0  8.0  9.0 10.0
MATRIZ RESULTADØ
  1.0  4.0  9.0 16.0 25.0
 36.0 49.0 64.0 81.0100.0
PRØMEDIØ   5.50
SUBTRACE *RETURN*
```

# APENDICE A

## FUNCIONES ESTANDAR

| Función General | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | Definición | Argumento(s) | | | Resultado | |
| | | Nº | Tipo | Rango | Tipo | Rango |
| **Logaritmo natural** | | | | | | |
| ALØG | $y = \log_e x$ | 1 | REAL*4 | $x>0$ | REAL*4 | $a \leq y \leq b$ |
| DLØG | ó $y = \ln x$ | 1 | REAL*8 | | REAL*8 | |
| **Logaritmo decimal** | | | | | | |
| ALØG10 | $y=\log_{10} x$ | 1 | REAL*4 | $x>0$ | REAL*4 | $c \leq y \leq d$ |
| DLØG10 | ó $y=\log x$ | 1 | REAL*8 | | REAL*8 | |
| **Exponencial** | | | | | | |
| EXP | $y=e^x$ | 1 | REAL*4 | $a \leq x \leq b$ | REAL*4 | $o \leq y \leq e$ |
| DEXP | | 1 | REAL*8 | | REAL*8 | |
| **Raíz cuadrada** | | | | | | |
| SQRT | $y = \sqrt{x}$ | 1 | REAL*4 | $x \geq 0$ | REAL*4 | $o \leq y \leq e^{1/2}$ |
| DSQRT | ó $y=x^{1/2}$ | 1 | REAL*8 | | REAL*8 | |
| **Arco seno** | | | | | | |
| ARSIN | $y=\text{arcsen}\,x$ | 1 | REAL*4 | $\lvert x \rvert \leq 1$ | REAL*4[-] | $-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$ |
| DARSIN | | 1 | REAL*8 | | REAL*8[-] | |
| **Arco coseno** | | | | | | |
| ARCOS | $y=\arccos x$ | 1 | REAL*4 | $\lvert x \rvert \leq 1$ | REAL*4[-] | $o \leq y \leq \pi$ |
| DARCOS | | 1 | REAL*8 | | REAL*8[-] | |

APENDICE A (Continuación)

FUNCIONES ESTANDAR (Continuación)

| Función General | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | Definición | Argumento(s) | | | Resultado | |
| | | Nº | Tipo | Rango | Tipo | Rango |
| **Arco tangente** | | | | | | |
| ATAN | y=arctgx | 1 | REAL*4 | Cualq. | REAL*4 | $-\frac{\pi}{2} \leq y$ |
| DATAN | | 1 | REAL*8 | real | REAL*8 | $\leq \frac{\pi}{2}$ |
| ATAN2 | y=arctg $\frac{x1}{x2}$ | 2 | REAL*4 | id.ex | REAL*4 | $-\pi \leq y$ |
| DATAN2 | | 2 | REAL*8 | cepto (0,0) | REAL*8 | $\leq \pi$ |
| **Seno** | | | | | | |
| SIN | y=senx | 1 | REAL*4 | $\lvert x\rvert < 2^{18} * \pi$ | REAL*4 | $-1 \leq y$ |
| DSIN | | 1 | REAL*8 | $\lvert x\rvert < 2^{50} * \pi$ | REAL*8 | $\leq 1$ |
| **Coseno** | | | | | | |
| COS | y=cos x | 1 | REAL*4 | $\lvert x\rvert < 2^{18} * \pi$ | REAL*4 | $-1 \leq y$ |
| DCOS | | 1 | REAL*8 | $\lvert x\rvert < 2^{50} * \pi$ | REAL*8 | $\leq 1$ |
| **Tangente** | | | | | | |
| TAN | y=tg x | 1 | REAL*4 | $\lvert x\rvert < 2^{18} * \pi$ | REAL*4 | $-e \leq y$ |
| DTAN | | 1 | REAL*8 | $\lvert x\rvert < 2^{50} * \pi$ | REAL*8 | $\leq e$ |
| **Cotangente** | | | | | | |
| COTAN | y=cotg x | 1 | REAL*4 | $\lvert x\rvert < 2^{18} * \pi$ | REAL*4 | $-e \leq y$ |
| DCOTAN | | 1 | REAL*8 | $\lvert x\rvert < 2^{50} * \pi$ | REAL*8 | $\leq e$ |

APENDICE A (Continuación)

FUNCIONES ESTANDAR (Continuación)

| Función General | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | Definición | Argumentos(s) | | | Resultado | |
| | | Nº | Tipo | Rango | Tipo | Rango |
| **Seno hiperbólico** | | | | | | |
| SINH | $y=\frac{e^{x}-e^{-x}}{2}$ | 1 | REAL*4 | $\lvert x\rvert<$ | REAL*4 | $-e\leq y$ |
| DSINH | | 1 | REAL*8 | 175.366 | REAL*8 | $\leq e$ |
| **Coseno hiperbólico** | | | | | | |
| COSH | $y=\frac{e^{x}+e^{-x}}{2}$ | 1 | REAL*4 | $\lvert x\rvert<$ | REAL*4 | $1\leq y$ |
| DCOSH | | 1 | REAL*8 | 175.366 | REAL*8 | $\leq e$ |
| **Tangente hiperbólica** | | | | | | |
| TANH | $y=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ | 1 | REAL*4 | Cualq. | REAL*4 | $-1\leq y$ |
| DTANH | | 1 | REAL*8 | real | REAL*8 | $\leq 1$ |
| **Valor absoluto** | | | | | | |
| IABS | $y=\lvert x\rvert$ | 1 | INTEGER*4 | entero | INTEGER*4 | |
| ABS | | 1 | REAL*4 | Cualq. | REAL*4 | |
| DABS | | 1 | REAL*8 | real | REAL*8 | |
| **Función error** | | | | | | |
| ERF | $y=\frac{2}{\sqrt{\pi}}\int_{0}^{x}e^{-u^{2}}du$ | 1 | REAL*4 | Cualq. | REAL*4 | $-1\leq y$ |
| DERF | | 1 | REAL*8 | real | REAL*8 | $\leq 1$ |

APENDICE A (Continuación)

FUNCIONES ESTANDAR (Continuación)

| Nombre | Definición | Argumento(s) | | | Resultado | |
|---|---|---|---|---|---|---|
| | | Nº | Tipo | Rango | Tipo | Rango |
| **Función General** | | | | | | |
| ERFC | $y=\frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-u^2}du$ | 1 | REAL*4 | Cualq. | REAL*4 | $0\leq y$ |
| DERFC | | 1 | REAL*8 | real | REAL*8 | $\leq 2$ |
| **Función gama y logaritmo de gama** | | | | | | |
| GAMMA | $y=\int_0^{\infty} u^{x-1}e^{-u}du$ | 1 | REAL*4 | $2^{-252}<x$ | REAL*4 | $g\leq y$ |
| DGAMMA | | 1 | REAL*8 | $< f$ | REAL*8 | $\leq e$ |
| ALGAMA | $y=\log_e\int_0^{\infty} u^{x-1}e^{-u}du$ | 1 | REAL*4 | $0<x$ | REAL*4 | $i\leq y$ |
| DLGAMA | | 1 | REAL*8 | $< h$ | REAL*8 | $\leq e$ |
| **Valores máximo y mínimo** | | | | | | |
| MAXO | $y=\max(x_1,\ldots,x_n)$ | ≥2 | INTEGER*4 | Cualq. | INTEGER*4 | |
| AMAXO | | ≥2 | INTEGER*4 | entero | REAL*4 | |
| MAX1 | | ≥2 | REAL*4 | Cualq. | INTEGER*4 | |
| AMAX1 | | ≥2 | REAL*4 | real | REAL*4 | |
| DMAX | | ≥2 | REAL*8 | | REAL*8 | |
| MINO | $y=\min(x_1,\ldots,x_n)$ | ≥2 | INTEGER*4 | Cualq. | | |
| AMINO | | ≥2 | INTEGER*4 | entero | | |
| MIN1 | | ≥2 | REAL*4 | Cualq. | | |
| AMIN1 | | ≥2 | REAL*4 | real | | |
| DMIN1 | | ≥2 | REAL*8 | | | |

APENDICE A (Continuación)

FUNCIONES ESTANDAR (Continuación)

| Función General | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | Definición | Argumento(s) | | | Resultado | |
| | | Nº | Tipo | Rango | Tipo | Rango |
| **Truncación** | | | | | | |
| AINT | y=(signo de | 1 | REAL*4 | | REAL*4 | |
| DINT | x) *n | 1 | REAL*8 | Cualq. | REAL*8 | |
| INT | donde n es el | 1 | REAL*4 | real | INTEGER*4 | |
| IDINT | mayor entero $\leq \lvert x \rvert$ | 1 | REAL*8 | | INTEGER*4 | |
| **Conversión a punto flotante** | | | | | | |
| FLOAT | Convierte de | 1 | INTEGER*4 | Cualq | REAL*4 | |
| DFLØAT | entero a real | 1 | INTEGER*4 | entero | REAL*8 | |
| **Conversión a punto fijo** | | | | | | |
| IFIX | Convierte de | 1 | REAL*4 | Cualq. | INTEGER*4 | |
| HFIX | real a entero | 1 | REAL*4 | real | INTEGER*2 | |
| **Transferencia de signo** | | | | | | |
| ISIGN | y=(signo $x_2$) | 2 | INTEGER*4 | entero | INTEGER*4 | |
| SIGN | $*x_1$ | 2 | REAL*4 | cualq. | REAL*4 | |
| DSIGN | $x_1 \neq 0$ | 2 | REAL*8 | real | REAL*8 | |

APENDICE A (Continuación)

FUNCIONES ESTANDAR (Continuación)

| Función General | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | Definición | Argumento(s) | | | Resultado | |
| | | Nº | Tipo | Rango | Tipo | Rango |
| Diferencia positiva | | | | | | |
| IDIM | $y=X_1-$mín | 2 | INTEGER*4 | entero | INTEGER*4 | |
| DIM | $(X_1,X_2)$ | 2 | REAL*4 | cualq. | REAL*4 | |
| DDIM | | 2 | REAL*8 | real | REAL*8 | |
| Parte más significativa de un argumento real | | | | | | |
| SNGL | | 1 | REAL*8 | real | REAL*4 | |

a = -180.218 b= 174.673 c= -78.268 d= 75.859

e = $16^{63}*(1-16^{-6})$ para REAL*4 y $16^{63}*(1-16^{-14})$para REAL*8

f = 257.5744 g= 0.88560 h= $4.2913*10^{73}$

i = -012149

# APENDICE B

## PROPOSICIONES Y CARACTERISTICAS DE FORTRAN IV COMPLETO QUE NO SON ACEPTADAS POR FORTRAN IV BASICO

ASSIGN

Arreglos con códigos de formato

BLOCK DATA*

COMPLEX

Constantes complejas, lógicas, literales y hexadecimales

Códigos de formato L,G y Z

COMMON con nombre

DATA

Dimensionamiento en tiempo de ejecución

ENTRY

Especificación de longitud en proposiciones de especificación de tipo

Facilidades de depuración

GO TO asignado

IMPLICIT

Inicialización de variables en proposiciones de especificación explícitas

IF lógico

Llamada por nombre

Literales como parámetros actuales

LOGICAL

Más de tres dimensiones en un arreglo

NAMELIST

Parámetros ERR y END en la proposición READ

PAUSE con literal

PRINT b, lista

PUNCH b, lista

READ b, lista

RETURN i con i distinto de blanco

Subíndices generalizados

* Las proposiciones y características subrayadas no han sido tratadas en este manual.

## APENDICE C

Anderson, Decima M., Computer Programming Fortran IV, Appleton-Century-Crofts/ Meredith Corporation, Estados Unidos, 1966, 435 pgs.

Dimitry, Donald y Mott, Thomas Jr., Introduction to Fortran IV Programming Holt, Rinehart and Winston, Inc., Estados Unidos, 1966, 334 pgs.

Hull, F.E., y Day, D.D, Computer and Problem Solving, Addison-Wesley, Estados Unidos, 1970, 276 págs.

IBM System Products Division, IBM System/360 and System/370 Fortran IV Language, Nueva York, 1971, 159 págs.

IBM System Products Division, IBM System/360 Disk Operating System Fortran IV Programmer's Guide, Nueva York, 1968, 96 págs.

IBM System Products Division, IBM System/360 Operating System Fortran IV (G and H) Programmer's Guide, Nueva York, 1968.

Nydegger, Adolph C., An Introduction to Computer Programming with an Emphasis on Fortran IV, Addison-Wesley, Estados Unidos, 1968, 268 págs.

Sánchez C., Víctor, Apuntes de Fortran, Santiago, Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, 1967, 77 págs.