SUBPROGRAMAS Y POSTULADOS AVANZADOS DEL FORTRAN.

INTRODUCCION.

Normalmente se espera que un programador profesional escriba programas complejos. Afortunadamente, dichos programas están formados de varios programas menos complicados. Ejemplo: para resolver un sistema de ecuaciones lineales simultáneas, por el método matricial, el problema se puede reducir a varias partes como son:

* LECTURA DE DATOS
* INVERSION DE UNA MATRIZ
* MULTIPLICACION DE MATRICES
* IMPRESION DE RESULTADOS

La complejidad del programa se reduce considerablemente si cada segmento del programa, se escribe, compila y prueba como una entidad separada. En esta unidad se describe la manera en que se puede definir y usar subprogramas que pueden compilarse y probarse independientemente del programa principal y que puede ser utilizado en P.P. de una manera tan fácil como si fueran funciones de biblioteca.

OBJETIVOS.

1. Saber que tipos de subprogramas puede construir.

2. Conocer las características de los subprogramas.

3. Saber las reglas para definir los subprogramas.

   a. De postulados aritmético
   b. Subprograma Function
   c. De subrutina

4. Conocer las reglas para dimensionar los arreglos en el uso de subprogramas Function y Subrutina.

5. Saber usar los subprogramas anteriores.

FORTRAN ADICIONAL. (Postulados Avanzados)

Objetivos :

1. Saber direccionar y almacenar información que ha de ser usada en las diferentes partes que forman un programa.

2. Saber como proporcionar los valores iniciales en un programa.

3. Conocer como accesar archivos definidos por el programador.

4. Saber manejar datos alfanuméricos, numéricos y lógicos en E/S.

En general conocer otros postulados que mejoran la eficiencia en la programación entre los cuales están los siguientes :

COMMON
EXTERNAL
COMMON/X/A, B, C, etc.
DATA/LISTA/VALOR 1, VALOR 2.
DEFINE FILE
READ a disco
WRITE a disco
LOGICAL
CHARACTER

PROCEDIMIENTO DE APRENDIZAJE.

Estudiar el anexo I, y para Fortran Adicional

° Estudiar el capítulo Nº 2 del libro

° Consultar las secciones V, VI, VII, XI, XII del

HP 3000 COMPUTER SYSTEMS
FØRTRAN Reference Manual

EXAMEN DE AUTOEVALUACION.

Puedes pedir tu examen de evaluación si puedes contestar las preguntas siguientes :

1.- Que es un paso de Parametros Argumentos del Programa.
2.- Cuantos tipos hay Subs

1. ¿Qué entiendes por salida de un subprograma?

2. ¿Cuántos tipos de subprogramas hay?

3. ¿En qué criterios te basas para seleccionar un subprograma?

4. Expresar con tus propias palabras las reglas para definir el subprograma de postulado aritmético.

   a. Nombre.
   b. Parte del programa en que se coloca.
   c. Número de postulados que lo definen.
   d. Distintas posibilidades como argumentos en la definición.
   e. Significado de los argumentos en relación al programa principal.
   f. Uso de estos subprogramas.

5. Resolver los siguientes problemas :

   En un problema determinado se requiere calcular varias veces las siguien-- tes funciones :

$$y(x) = x^3 + 5x^2 - \ln(x + 1.0) + 12$$

$$y'(x) = -3x^2 + 10x - \frac{1}{x + 1}$$

$$y''(x) = -6x + 10 + \frac{1}{(x + 1)^2}$$

   Hacer los subprogramas del postulado aritmético, llamando a :

y	⟶	FX
y'	⟶	DFX
y''	⟶	SDFX

6. Expresar con tus propias palabras las reglas para definir un subprograma - FUNCTION con relación a :

   a. Nombre.

   b. En qué parte del programa se pone.

   c. Número de instrucciones permitidas para su definición.

d. Qué significado tienen los argumentos en la definición con relación al programa principal.

e. Cuando se usan arreglos en el subprograma, reglas que se deben de respetar con relación al dimensionado de los arreglos del programa principal.

f. Cuando se usan arreglos en el subprograma, reglas que se deben de respetar con relación al dimensionado de los arreglos del programa principal.

g. Uso de subprogramas.

h. Distintas posibilidades como argumento para su uso.

i. Para que se usa RETURN.

7. Resolver los siguientes problemas:

a. Escribir un subprograma FUNCTION para que dado un arreglo llamado A de 2 dimensiones y con el mismo número de filas y de columnas, nos dé como salida la suma de los cuadrados de los productos que se encuentran en la fila M, y en la columna N es decir si el arreglo tiene NF = número de filas o de columnas. El programa se puede emplear hasta un arreglo de 20 x 20.

b. Usar el programa anterior para un arreglo llamado B con los siguientes datos:

B =

1.1	6.5	5.1	6.8	3.1
7.8	1.2	6.3	9.4	10.2
9.6	7.8	1.3	7.6	8.5
9.4	7.8	9.6	1.4	6.5
6.5	7.6	5.4	9.8	1.5

para la fila 3 y la columna cuatro y el resultado guardarlo en S34.

Para la fila I y para la columna J definiendo previamente los valores de I y de J.

8. Expresar con tus propias palabras las reglas para definir un subprograma de SUBRUTINA.

a. Nombre.

A = B + C
Matrices

b. Lugar del programa en que se coloca.

c. Distintas posibilidades como argumentos en la definición.

9. Explicar con tus propias palabras las reglas para definir una subrutina con relación a:

a. Nombre.

b. Parte del programa en donde se coloca.

c. Número de instrucciones permitidas para su definición.

d. Distintas posibilidades como argumento en la definición.

e. Significado que tienen los argumentos en la definición con relación al programa principal.

f. Cuando se usan arreglos en la subrutina, reglas que se deben respetar con relación al dimensionado de los arreglos del programa principal.

g. Uso a la subrutina.

h. Distintas posibilidades como argumentos para su uso.

10. Resolver los siguientes problemas:

a. Dado un arreglo A de una dimensión hacer una subrutina que determine la suma de los valores positivos y la suma de los valores negativos, el número de elementos que son mayores que cero y el número de elementos que son menores que cero.

b. Hacer una subrutina para ordenar los valores numéricos de un arreglo C. en orden ascendente o descendente dependiendo de una variable clave N. Si la variable es cero, los elementos se arreglan en orden ascendente si el valor de la variable es diferente de cero se deben ordenar en forma descendente.

REQUISITO:

Antes de presentar el examen de evaluación, deberás entregar un problema resuelto por computadora en donde uses todo lo aprendido en la unidad.

ANEXO I.

FUNCIONES Y SUBRUTINAS :

Hasta ahora se han empleado ciertas funciones cuyos nombres en FØRTRAN son :

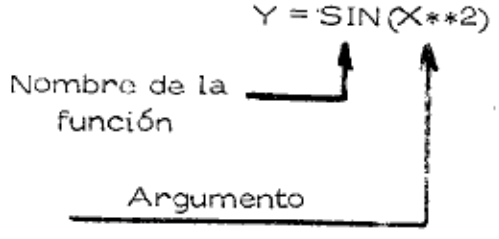
SIN	ABS	SQRT	*FLOAT
CØS	ALØG	EXP	*BOOL
ATAN	ALØG 10	*I FIX	*CMPLX

*Consultar Manual HP3000, página 163.

Y para usarlos es suficiente con nombrarlos y poner entre paréntesis el argumento.

Ejemplo.

Si se desea obtener el sen (X^2) y el resultado asignarlo a una variable que se llame Y se tiene la siguiente expresión :



Si se desea hacer $X = \left| X1^2 - Y1^2 \right|$ es suficiente con poner :

X = ABS (X1**2 - Y1**2)

En este caso se determina el valor absoluto de la expresión encerrada entre paréntesis, es decir X1**2 - Y1**2 y el resultado se asigna a la variable X.

La forma general de esta función es :

Variable = Expresión que puede contener funciones.

La función forma parte de la expresión.

Ejemplo.

$$YP = \sqrt{X^2 - Y^2}$$

```
YP = SQRT (ABS (X**2 - Y**2))
```

$$Y = \ln \left| X^2 - Y^2 \right|$$

```
Y = ALØG (ABS (X**2 - Y**2))
```

Tanto el argumento como el resultado de una función son números reales.

Observaciones :

No puede aparecer una función en el lado izquierdo de un postulado de asignación aritmética.

Cada vez que usamos una función, se obtiene un resultado.

Ejemplos :

CNT = FLOAT (KNT)	KNT = IFIX (CNT)	Y = SIN(X)
el valor de la variable KNT se convierte de entero a real y se asigna a la posición CNT.	el valor de la variable CNT se convierte de real a entero y se asigna a KNT.	Argumento = X Función Seno Resultado Sen(X)

y el valor se asigna a la variable Y.

Cuando se habla de que se obtiene un resultado, también se dice que se tiene una salida.

El programador no tiene control sobre las funciones anteriores.

Para usar una función dada, sólo se requiere escribir el nombre de la función seguida por una expresión encerrada en paréntesis a la que se le llama argumento.

Dos preguntas nos podemos hacer en este momento :

¿Sería conveniente que pudiéramos inventar o diseñar cierto tipo de funciones y subprogramas que si tuvieramos control de ellos ?

Y ¿qué tipos de programas podemos diseñar ?

Las respuestas son las siguientes.

1ª Sí podemos diseñar ciertos tipos de funciones y subprogramas y

2º Hay 3 tipos de subprogramas que se pueden diseñar por el programador, es decir programas sobre los cuales tiene control el programador. Y son:

a. Subprogramas de proposición aritmética.
b. Subprograma FUNCTION.
c. Subrutinas.

Para determinar qué tipo de subprograma es apropiado, se debe primero:

Determinar el número de proposiciones requeridas para definir el subprograma - y entonces determinar el número de valores de salida que el subprograma debe - obtener en cada ejecución.

Si se desea más de un valor de salida para cada ejecución del subprograma, se - debe usar una subrutina.

El subprograma de proposición aritmética y el subprograma FUNCTION se limi— tan a un valor de salida.

Si este valor se puede definir por una proposición aritmética entonces se debe - emplear el subprograma de proposición aritmética. Si varias proposiciones (arit méticas y otros) se necesitan para definir el valor, entonces se debe emplear el subprograma FUNCTION

SELECCION DE SUBPROGRAMAS.

La selección del subprograma se basa en:

a. Número de postulados necesarios para definir el subprograma.

b. Número de valores de salida que el subprograma debe proporcionar en cada ejecución.

En las funciones que se han empleado, se ha asignado un nombre determinado a - cada función y dicho nombre está fuera del control del programador. También a - los subprogramas y a las funciones se les debe asignar un nombre, para lo cual, se deben respetar ciertas reglas.

Para subprogramas de proposición aritmética y subprogramas FUNCTION, como se tiene un valor de salida, el nombre se les asigna de acuerdo con las reglas pa— ra asignar nombre a las variables, es decir:

Se emplea de 1 a 6 caracteres alfanuméricos (letras o dígitos), el primero de los cuales es una letra.

La primera letra deberá ser I, J, K, L, M o N. Si el valor de salida del sub- - programa es una cantidad entera y diferente de las mencionadas si el valor de - salida es un número real.

Para el caso de las subrutinas, como se pueden tener varios valores de salidas (enteros y reales), no hay ninguna restricción sobre la primera letra del nom- - bre.

Para cada tipo de subprograma se debe tener en claro dos cosas :

1º Definir el subprograma.
2º Usar el subprograma.

SUBPROGRAMAS DE PROPOSICION ARITMETICA.

Supóngase que la ecuación $y = \frac{(e^x + e^{-x})}{2} \cos(x) - 1$ se emplea repetidamente - en un programa. Se desea producir un subprograma que tenga como salida un - solo valor (Y).

Este subprograma de postulado aritmético se puede formar de la siguiente mane_ ra :

```
Y(X) = (EXP(X) + EXP(-X)) * CØS(X)/2.0 - 1.0
```

Nombre de la función

Argumento de la función

Proposición aritmética que define la función

El nombre de la función es Y. (La salida será un número real).

El argumento es X (puede haber varios argumentos, en cuyo caso se encierran - entre paréntesis y se separan por comas).

Las variables con subíndice no se permiten como argumentos en la definición.

El postulado aritmético que define a la función es :

```
(EXP(X) + EXP(-X)) * CØS(X)/2.0 - 1.0
```

Variables no incluidas como argumentos pueden aparecer en el postulado de definición.

Funciones previamente definidas o subprogramas pueden aparecer en la definición.

La proposición que define el subprograma debe aparecer al frente del subprograma antes que cualquier proposición ejecutable en el programa.

El subprograma y los nombres de los argumentos se pueden considerar como independientes del programa principal.

El nombre de X tiene significado en el subprograma solamente y se usa para definir la manera en que se llevan a cabo los cálculos (la regla de correspondencia). Son variables mudas. Si el programa principal tiene una variable que se llama X, este nombre no se relaciona con el nombre X del subprograma.

Si en la definición de un subprograma no se permiten los argumentos con subíndices. Esto no se debe considerar como una limitación, ya que los argumentos en la definición son variables mudas que no tienen ninguna relación con el programa principal. Posteriormente cuando el programa se usa, un conjunto de argumentos sustituirán a los argumentos mudos.

Estos argumentos serán variables si están asociados con el programa principal y estos argumentos sí pueden ser variables con subíndice.

MIENTRAS QUE LAS VARIABLES CON SUBINDICE NO SE PERMITEN COMO ARGUMENTOS EN LA DEFINICION DE UN SUBPROGRAMA, SI SE PERMITEN EN EL USO.

USO DE LOS SUBPROGRAMAS DE PROPOSICION ARITMETICA.

Para usar un subprograma de proposición aritmética se escribe el nombre del subprograma y, en lugar de las variables mudas (que se usaron como argumentos en la definición) se sustituyen variables, constantes o expresiones que tengan significado en el programa principal.

Como ejemplos supongan que se desea determinar el valor de la función para los siguientes casos :

$$A = \left(\left(\frac{e^{(S1)} + e^{(-S1)}}{2.0}\right) \cos(S1) - 1.0\right) +$$

$$\left(\left(\frac{e^{(S1+DX)} + e^{(-(S1+DX)}}{2.0}\right) \cos(S1+Dx) - 1.0\right) * DX/2.0$$

o sea determinar el valor de Y en el punto S1, determinar el valor de Y en el punto S1 + DX. Sumar los valores anteriores y el resultado multiplicarlo por DX y dividirlo por 2.

La expresión anterior se puede obtener de la siguiente forma:

$$A = (Y(S1) + Y(S1 + DX)) * DX / 2.0$$

también se podría tener una expresión como la siguiente:

$$A1 = (Y(A(5)) + Y(S(I))) / Y(ABS(X ** 2 - Y1 ** 2))$$

observa que se emplean como argumentos variables con subíndice A y S y una función conocida ABS.

Ejemplo:

Empleando la regla de Simpson determinar el area bajo la curva:

$$Y = \frac{e^{x^2}}{x^2} \quad \text{donde} \quad XI = 0.5 \quad y \quad XF = 2.5$$

dividiendo el intervalo en 50 partes.

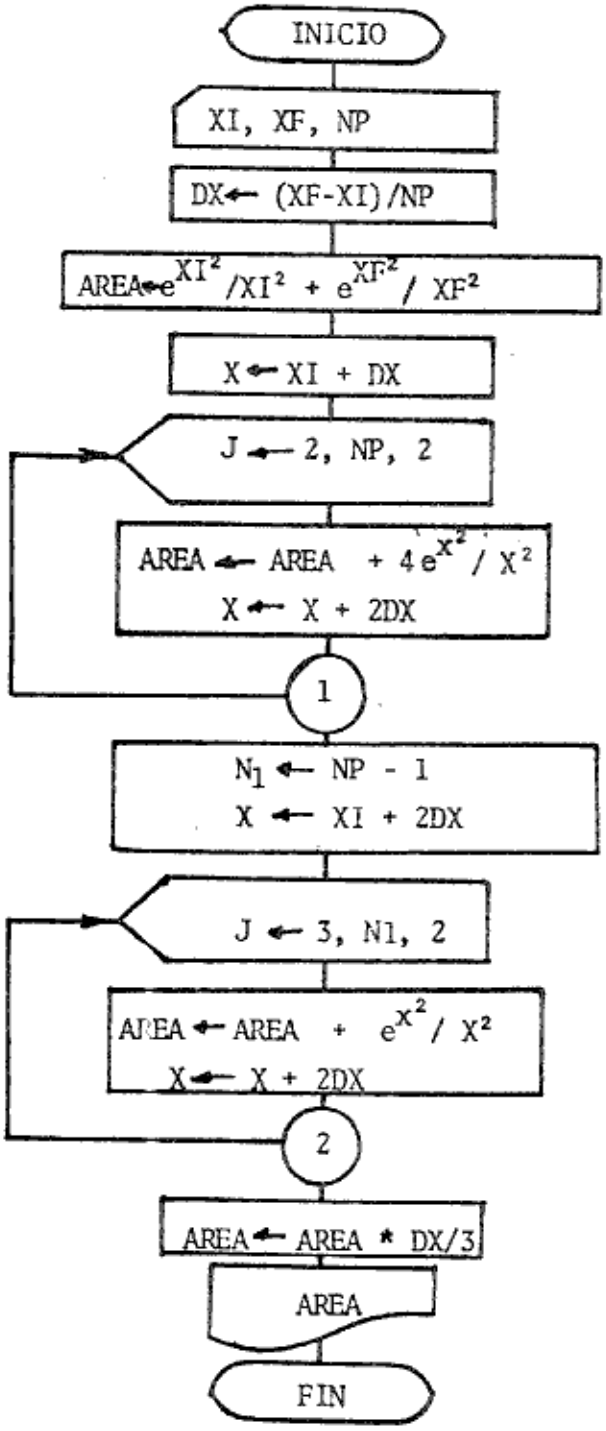
Hacer el diagrama de flujo y programa para la computadora digital.

a. Sin emplear subprogramas.
b. Empleando subprogramas.

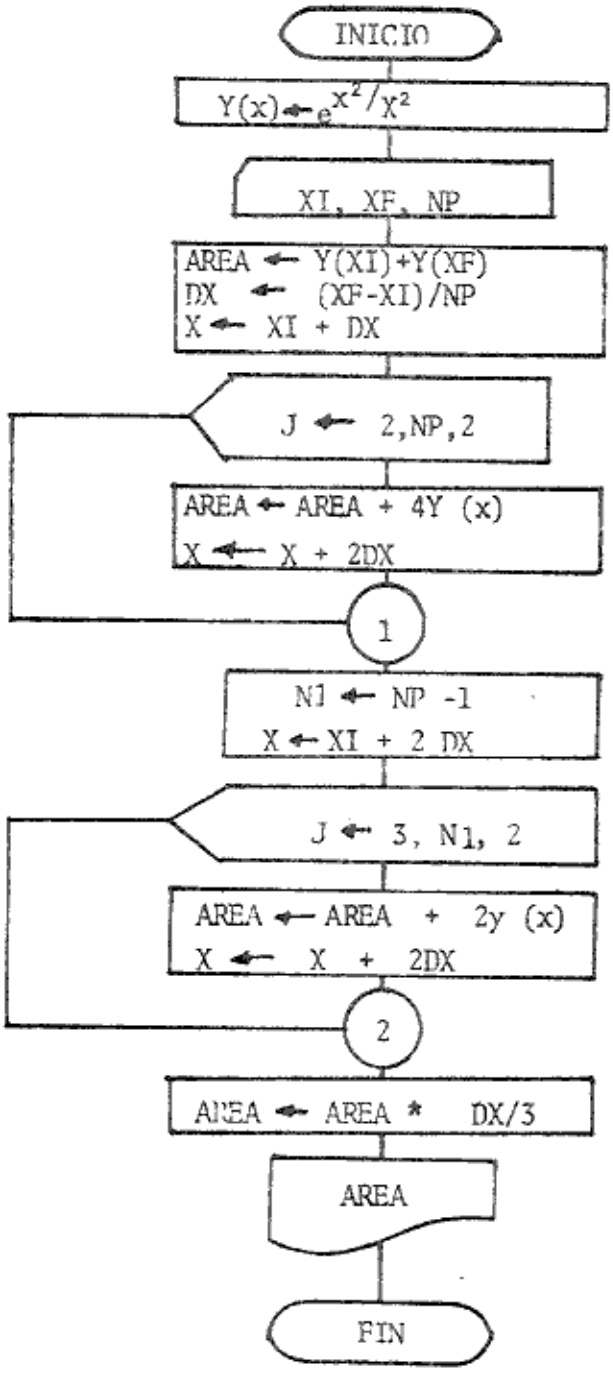
La fórmula para determinar el area bajo una curva por la regla de Simpson es:

$$\text{Area} = \frac{\Delta X}{3} \left(Y_1 + 4 \sum_{i=2,4,6}^{i=n} Y_i + 2 \sum_{i=3,5,7}^{i=n-1} Y_i + Y_{n+1}\right)$$

a. Diagrama de flujo sin emplear subprogramas.



b. Diagrama de flujo empleando subprograma



La codificación es la siguiente :

```
TARJETAS DE CONTROL
      6
        Y(X)=EXP(X**2)/X**2
        READ(5,8)XI,XF,NP
    8   FØRMAT(2F8.3,I3)
        AREA=Y(XI)+Y(XF)

        DØ 20 J=2,NP,2
        AREA=AREA+4.0*Y(X)
        X=X+2.0*DX
   20   CØNTINUE
        N1=NP-1
        X=XI+2.0*DX
        DØ 30 J=3,NI ,2
        AREA=AREA+2.0*Y(X)
        X=X+2.0*DX
   30   CØNTINUE
        AREA=AREA*DX/3.0
        WRITE(6,8)AREA
    8   FØRMAT(5X,'EL AREA ES',E13.6)
        STOP
        END
TARJETAS DE CONTROL
        DATØS
```

SUBPROGRAMA FUNCTION.

Este tipo de subprograma se tiene un valor de salida V (cada vez que se usa se obtiene un resultado) y para definirlo se requiere de varias instrucciones que pueden ser aritméticas y lógicas.

Definición:

Para definir un subprograma FUNCTION se requiere en primer lugar, la palabra FUNCTION, seguida por el nombre de la función (ya se mencionaron las reglas que deben seguir para asignar el nombre) y finalmente la lista de argumentos de la función, encerrados entre paréntesis y separados por comas.

En este caso, igual que en el anterior no se permite como argumento las variables con subíndice, además se debe indicar donde termina el subprograma para lo cual empleamos la proposición END. Es necesario también transferir el control al programa principal por distintas causas, para lo cual se emplea la proposición RETURN. Es posible el uso de varias proposiciones RETURN. Finalmente se requiere que cuando menos una vez aparezca el nombre del subprograma de lado izquierdo de una proposición de asignación aritmética o bien se le asigne un valor por medio de una proposición de lectura con el objeto de que la función obtenga un valor cada vez que se usa.

Ejemplo:

Dada una lista de números enteros, hacer un subprograma FUNCTION para determinar la suma de los elementos elevados a una potencia I, se podrá usar hasta con 50 elementos.

Los argumentos en este caso son:

a. Nombre del arreglo de la lista de los elementos
b. Tamaño del arreglo y
c. Exponente al que se eleva cada elemento.

Se puede emplear el nombre SUM para identificar a la función y el subprograma sería como sigue:

```
      6
      |FUNCTIØN SUM(A,N,I)
      |DIMENSIØN A(50)
      |SUM=0.0
      |DØ 20 J=1,N
   20 |SUM=SUM+A(J)**I
      |RETURN
      |END
```

Observaciones :

El nombre del subprograma es SUM

A representa el nombre del arreglo en donde se guarda la lista de números y el tamaño máximo es 50.

N representa el tamaño del arreglo.

I es el exponente al que se eleva cada elemento.

En dos instrucciones aparece el nombre de la función al lado izquierdo de una proposición de asignación aritmética.

```
SUM = 0.0
```

y

```
SUM = SUM + A(J) ** I
```

Se usa el postulado

```
RETURN
```

y

```
END
```

USO DE LOS SUBPROGRAMAS FUNCTION :

Supongan por ejemplo que se tiene un arreglo B donde :

B =

3.0
2.5
4.6
7.4
9.8
5.4

y se desea obtener el siguiente valor :

$$S = \sum_{i=1}^{6} B_i^2$$

o sea la suma de los cuadrados de los elementos de B asignarlos a la variable S.

Usando la función SUM, se tendrían como argumentos B, el tamaño en este caso es 6, y el exponente 2.

$$S = SUM(B, 6, 2)$$

Significa que se desea que ejecute las instrucciones indicadas en el subprograma que se llama SUM, que en lugar de A utilice el arreglo B, en lugar de N, use el número 6 y que en lugar de I + 1, use el número 2.

Otras formas admitidas podrían ser

$$S1 = SUM(X, A(I), I+1)$$

En este caso se tiene como argumento una variable con subíndice, y una expresión aritmética.

Supongan ahora que se desea obtener los siguientes resultados

$$S1 = \sum_{i=1}^{6} B_i^2$$

$$S2 = \sum_{i=1}^{6} B_i^4$$

$$S3 = \sum_{i=1}^{6} B_i^5$$

$$P = \frac{S1 + S2}{S3}$$

El programa lo podríamos tener en la siguiente forma :

```
    6
      TARJETAS DE CØNTRØL
      DIMENSIØN B(50)
      READ(5,8)(B(I),I=1,6)
    8 FØRMAT(6F8.2)
      S1=SUM(B,6,2)
      S2=SUM(B,6,4)
      S3=SUM(B,6,5)
      P=(S1+S2)/S3
      WRITE(6,9)S1,S2,S3,P
    9 FØRMAT(5X,'S1=',E13.6,5X,'S2=',E13.7,5X,'S3=',E13.6,5X,'P=',E13.6)
      STOP
      END
      FUNCTIØN SUM(A,N,I)
      DIMENSIØN A(50)
      SUM=0.0
      DØ 20 J=1,N
   20 SUM=SUM+A(J)**I
      RETURN
      END
      TARJETAS DE CØNTRØL
      DATØS
```

SUBRUTINAS.

La subrutina es el subprograma más sofisticado. Puede tener su propia proposición de lectura. A diferencia de los otros subprogramas no hay restricción en el número de valores de salida. La definición de una subrutina se parece a la definición de un subprograma FUNCTION.

Un grupo de proposiciones se escriben precedidas por una proposición de identificación que contiene :

a. La palabra subrutina, seguida por :
b. El nombre de la subrutina seguida por
c. Los argumentos encerrados entre paréntesis y separados por comas.

Una proposición RETURN debe aparecer al menos una vez y la última proposición es una proposición END.

Puesto que la subrutina puede producir más de una salida, el primer caracter en el nombre de la subrutina no es una restricción.

Definición:

Para explicar la manera en que se define una subrutina se hará uso de un ejem-- plo.

Supongan que se desea hacer un subprograma para que:

Dado un arreglo de una dimensión de tamaño N, que nos representa una lista de números, se obtenga el número de elementos del arreglo que son menores que 6, el número de elementos del arreglo que son iguales que 6 y finalmente el número de elementos del arreglo que son mayores que 6.

En este caso, cada vez que se use el subprograma se deberá especificar el nombre del arreglo, el tamaño y se tendrá como resultados (salida) 3 números enteros.

Así se puede decir que las entradas de la subrutina son el nombre del arreglo y el tamaño, y su salida los 3 números enteros.

Llamando al subprograma XMM6 y considerando como nombre del arreglo para el argumento a A y el tamaño como N, MA6 variable para asignarle el número de elementos mayores que 6, ME6 variable para guardar el número de elementos iguales que 6. El subprograma sería como sigue:

```
      6
      |  SUBROUTINEXMM6(A, N, MA6, ME6, IG6)
      |  DIMENSION A (100)
      |  MA6 = 0
      |  ME6 = 0
      |  IG6 = 0
      |  DØ 20 I=1,N
      |  IF (A(I)-6.0)10, 11 , 12
   10 |  ME6=ME6+1
      |  GØ TØ 20
   11 |  IG6=IG6+1
      |  GØ TØ 20
   12 |  MA6=MA6+1
   20 |  CØNTINUE
      |  RETURN
      |  END
```

Las cantidades

A = Nombre del arreglo (Entrada)
N = Tamaño del arreglo (Entrada)
MA6 = Número de elementos mayores que 6 (Salida)
ME6 = Número de elementos menores que 6 (Salida)
IG6 = Número de elementos iguales que 6 (Salida)

Son argumentos de la subrutina y aparecen en la proposición de identificación.

Como en este caso se tienen 3 valores en la salida, es necesario incluir en este tipo de subprogramas los argumentos de salida (nombres de variables que identifican donde se van a guardar los valores de salida).

En los otros tipo de subprogramas, donde se tenía un solo valor de salida, se usó el nombre del subprograma para indicar donde se iba a almacenar el valor de salida.

Los argumentos en la definición pueden incluir nombre de arreglos, variables que sean suscritas.

Como antes, las variables con subíndice no pueden aparecer como argumentos en la definición. Pero pueden aparecer como argumentos cuando se emplea la subrutina. Esto no implica que las variables con subíndice no puedan aparecer dentro del subprograma. La restricción es solamente con respecto al argumento.

USO DE LAS SUBRUTINAS.

Para usar una subrutina es necesario escribir una proposición separada para llamarla (en el programa principal). Esto es, la subrutina no puede usarse simplemente mencionando su nombre, sino que se requiere una proposición para llamarla. Por ejemplo:

```
CALL XMM6 (B, 50, I1, I2, I3)

CALL XMM6 (R, A(I), NI, NA, MA)
```

Antes de que la subrutina puede ser empleada, se debe tener en el programa principal las proposiciones que permiten identificar un arreglo que contenga un máximo de 100 elementos. Lo que obliga a una proposición DIMENSIØN en el P.P., y también en el subprograma.

El nombre del arreglo en el programa principal y el nombre del arreglo en el subprograma son independientes y requieren proposiciones de DIMENSIØN independientes como se puede observar, el resultado de esto es un desperdicio de memoria.

Ejemplo.

Supóngase que se tiene un arreglo A de M filas y N columnas y que se desea hacer una subrutina para imprimir el arreglo de K en K columnas, indicando la fila y la columna con rótulos.

Por ejemplo: Si el arreglo A es de 50 filas y 36 columnas y se desea que imprima de 10 en 10 se tendría como salida

	1	2	3	...	10
1	A_{11}	A_{12}	A_{13}	...	$A_{1,10}$
2	A_{21}	A_{22}	A_{23}	...	$A_{2,10}$
3	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
50	$A_{50,1}$	$A_{50,2}$	$A_{50,3}$	...	$A_{50,4}$

	11	12	13	...	20
1	$A_{1,11}$	$A_{1,12}$	$A_{1,13}$	...	$A_{1,20}$
2	$A_{2,11}$	$A_{2,12}$	$A_{3,13}$	...	$A_{3,20}$
3	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
50	$A_{50,1}$	$A_{50,2}$	$A_{50,3}$	...	$A_{50,20}$

	21	22	23	. . .	30
1	$A_{1,21}$	$A_{1,22}$	$A_{1,23}$	. . .	$A_{1,30}$
2	$A_{2,21}$	$A_{2,22}$	$A_{2,23}$	. . .	$A_{2,30}$
3	$A_{3,21}$	$A_{3,22}$	$A_{3,23}$	. . .	$A_{3,30}$
.	.	.	.	. . .	.
.	.	.	.	. . .	.
.	.	.	.	. . .	.
50	$A_{50,21}$	$A_{50,22}$	$A_{50,23}$	. . .	$A_{50,30}$

	31	32	33	. . .	36
1	$A_{1,31}$	$A_{1,32}$	$A_{1,33}$	. . .	$A_{1,36}$
2	$A_{2,31}$	$A_{2,32}$	$A_{2,33}$	. . .	$A_{2,36}$
3	$A_{3,31}$	$A_{3,32}$	$A_{3,33}$	. . .	$A_{3,36}$
.	.	.	.	. . .	.
.	.	.	.	. . .	.
.	.	.	.	. . .	.
50	$A_{50,31}$	$A_{50,32}$	$A_{50,33}$	. . .	$A_{50,36}$

Explicación.

Los argumentos de la subrutina son:

a. Nombre del arreglo (A)
b. Número de filas del arreglo (NFA)
c. Número de columnas del arreglo (NCA)
d. Números de columnas que se imprimen (NCPS) en cada línea.

Nombre de la subrutina es ESCR.

El procedimiento es como sigue:

1º Se define un formato para el máximo número de columnas por ejemplo 10.

2º También se requiere un formato para los rótulos (formatos 173 y 176 de la codificación).

3º El procedimiento se va a repetir la parte entera de NCA/NCPS más uno.

En los primeros NCA/NCPS pasos se deben imprimir NCPS y en el último – NCA – NCA/NCPS.

En seguida se muestra la codificación donde:

A = Nombre del arreglo que se va a imprimir, en este caso se usa de 10×10.

I_2 = Arreglo donde se guardan los números para rótulo.

N10 = Número de veces que se realiza el proceso.

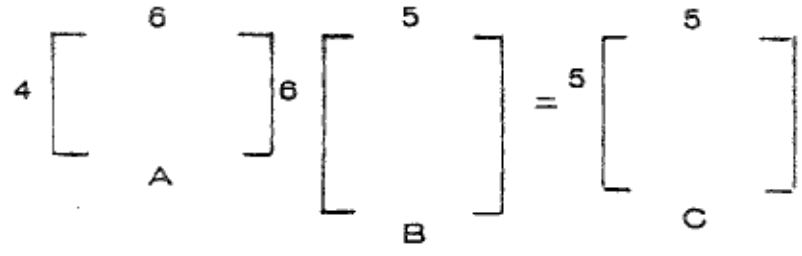
NIN y NFI = Variables enteras usadas para la generación de rótulos en los primeros N10 – 1 pasos.

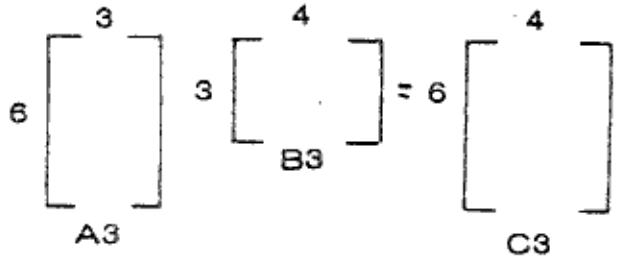
NT1 = Número de la última columna.

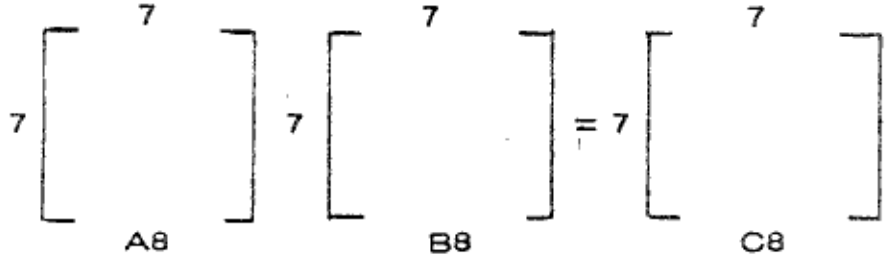
```
      6
        SUBROUTINE ESCR (NCPS,A,NFA,NCA)
        DIMENSION A (10,10),I2(10)
        IF(NCA-NCPS)13,14,13
   14   N10=1
        GØ TØ 15
   13   N10=NCA/NCPS+1
   15   DO 20 K=1,N10
        NIN=K*NCPS+1-NCPS
        NFI=K*NCPS
        I1=NIN-1
        NTI=NCA-(N10-1)*NCPS
        IF(K-N10)10,11,10
   10   DO 21 J=1,NCPS
        I2(J)=I1+J
   21   CONTINUE
  173   FORMAT(//10(10X,I3)//)
  176   FORMAT(//3X,I3,4X,10F12.4)
        WRITE(6,173)(I2(J),J=1,NCPS)
        GØ TØ 16
   11   DØ 22 J=1,NT1
        I2(J)=I1+J
   22   CONTINUE
        WRITE(6,173)(I2(J),J=1,NT1)
        NFI=NCA
   16   DØ 23 J=1,NFA
        WRITE(6,176)J,(A(J,L),L=NIN,NFI)
   23   CONTINUE
   20   CONTINUE
        RETURN
        END
```

Ejemplo :

Supóngase que en un programa determinado se desea multiplicar las siguientes - parejas de matrices :







El programa para multiplicar una matriz A de 3 filas y 4 columnas por una matriz B de 4 filas y 5 columnas y el resultado se guarda en una matriz C de 3 filas y 5 columnas es como sigue :

El objetivo es llenar los elementos de C de acuerdo con la siguiente fórmula :

$$C_{ij} = \sum_{l=1}^{4} A_{il} B_{lj}$$

i = 1 , 3

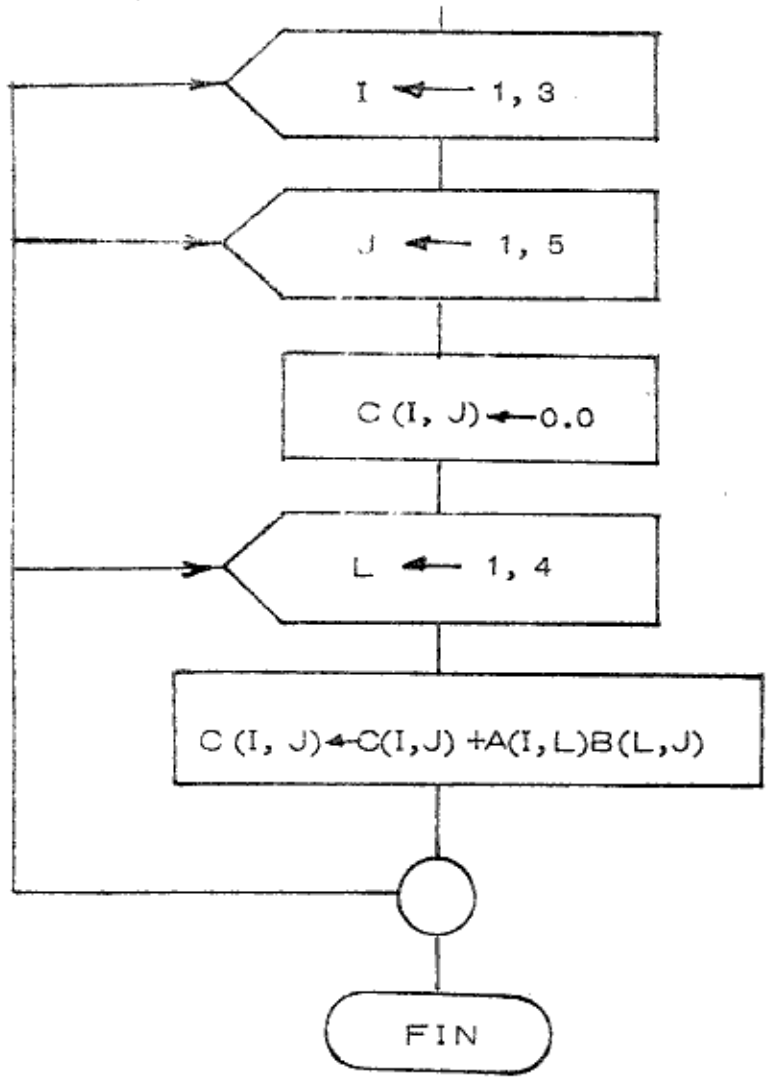
j = 1 , 5

un posible algoritmo sería :

Considerar que se llenan los elementos de C por filas o sea que se deberán llenar 3 filas, al llenar cada fila, considerar que se deberán llenar 5 columnas y para llenar cada elemento se deben efectuar 4 sumas de productos.

El diagrama de flujo considera un DØ para llenar las filas, dentro del DØ anteriós otro DØ para llenar las columnas y dentro del DØ anterior otro DØ para efectuar la suma de productos.

En seguida se muestra el diagrama de flujo :



```
      DIMENSIØN A(3,4),B(4,5),C(3,5)
      DØ 20 I=1,NFA
      DØ 30 J=1,NCB
      C(I,J)=0.0
      DØ 40 L=1,NCA
      C(I,J)=C(I,J)+A(I,L)*B(L,J)
   40 CØNTINUE
   30 CØNTINUE
   20 CØNTINUE
```

De acuerdo con lo anterior el programa para resolver las 3 multiplicaciones de matrices sería :

```
      6
      DIMENSIØN A(4,6),B(6,5),C(5,4),A3(6,3),B3(3,4),C3(6,4),
     1          A8(7,7),B8(7,7),C8(7,7)
      DØ 20 I=1,4
      DØ 30 J=1,5
      C(I,J)=0.0
      DØ 40 L=1,6
      C(I,J)=C(I,J)+A(I,L)*B(L,J)
   40 CØNTINUE
   30 CØNTINUE
   20 CØNTINUE
      DØ 50 I=1,6
      DØ 60 J=1,4
      C3(I,J)=0.0
      DØ 70 L=1,3
      C3(I,J)=C3(I,J)+A3(I,L)*B3(L,J)
   70 CØNTINUE
   60 CØNTINUE
   50 CØNTINUE
      DØ 80 I=1,7
      DØ 90 J=1,7
      C8(I,J)=0.0
      DØ 100 L=1,7
      C8(I,J)=C8(I,J)+A8(I,L)*B8(L,J)
  100 CØNTINUE
   90 CØNTINUE
   80 CØNTINUE
```

El problema anterior se hará haciendo uso de subrutinas.

Los argumentos de la subrutina son :

a. Matriz que premultiplica (D)

b. Matriz que postmultiplica (B)

c. Matriz donde se guarda el resultado (C)

d. Número de filas de la matriz que premultiplica (NFA)

e. Número de columnas de la matriz que premultiplica (NCA)

f. Número de columnas de la matriz que postmultiplica (NCB)

Se le puede asignar el nombre MUMAT y usando el diagrama de flujo conocido para multiplicar matrices, la codificación sería:

```
C                    DEFINICION
      SUBRØUTINE MUMAT(A,B,C,NFA,NCA,NCB)
      DIMENSIØN A(7,7),B(7,7),C(7,7)

      DØ 20 I=1,NFA
      DØ 30 J=1,NCB
      C(I,J)=0.0
      DØ 40 L=1,NCA
      C(I,J)=C(I,1)+A(I,L)*B(L,J)
   40 CØNTINUE
   30 CØNTINUE
   20 CØNTINUE
      RETURN
      END
```

```
C                    USO
      DIMENSION A(7,7),B(7,7),C(7,7),A3(7,7),B3(7,7),C3(7,7),
     1          A8(7,7),B8(7,7),C8(7,7).
            .
            .
            .
      CALL MUMAT (A,B,C,4,6,5)
      CALL MUMAT (A3,B3,C3,6,3,4)
      CALL MUMAT (A8,B8,C8,7,7,7)
            .
            .
            .
```

```
      6
        STOP
        END
        SUBROUTINE MULMAT (A,B,C,NFA,NCA,NCB)
        DIMENSION A(7,7),B(7,7),C(7,7)
        DØ 20 I=1,NFA
        DØ 30 J=1,NCB
        C(I,J)=0.0
        DØ 40 L=1,NCA
        C(I,J)=C(I,J)+A(I,L)*B(L,J)
   40   CØNTINUE
   30   CØNTINUE
   20   CØNTINUE
        RETURN
        END
        TARJETAS DE CONTROL
        DATOS
```

Observaciones :

a. Cuando no se usan subrutinas, en el postulado de DIMENSIØN, se dimensionan los arreglos de acuerdo a como se requiere.

b. Se repite el programa de multiplicar matrices cada vez que se requiere.

c. Cuando se usa la subrutina, en el postulado de DIMENSIØN, se dimensionan los arreglos de acuerdo a como se dimensionaron en la subrutina.

d. No se requiere repetir el programa tantas veces como se requiere, basta con llamar a la subrutina.

Ejemplo :

Hacer una subrutina para leer arreglos de 2 dimensiones.

Los argumentos son :

a. Nombre del arreglo que se desea leer (A).

b. Número de filas del arreglo que se desea leer (NFA).

c. Número de columnas del arreglo que se desea ver NCA.

En la subrutina se deberá pasar un formato que es el número de datos que se leerán en cada tarjeta por ejemplo 10F 8.3 es decir se pueden poner 10 datos en cada tarjeta.

El subprograma sería:

```
      SUBRØUTINE LEE (A , NFA , NCA)
      DIMENSION A (7 , 7)
      READ (5 , 8) ((A(I , J), J=1 , NCA) , I=1 , NFA)
8     FØRMAT (10F 8.3)
      RETURN
      END
```

Ejemplo.

Suponga que además de multiplicar las matrices del ejemplo anterior, se requiere también leer los datos e imprimir los resultados usando las subrutinas de MUMAT y ESCR.

El programa sería:

```
      6
      |STOP
      |END
      |SUBROUTINE MUMAT (A,B,C,NFA,NCA,NCB)
      |DIMENSIØN A(7,7),B(7,7),C(7,7)
      |DØ 20 I=I,NFA
      |DØ 30 J=1,NCB
      |C(I,J)=0.0
      |DØ 40 L=1,NCA
      |C(I,J)=C(I,J)+A(I,L)*B(L,J)
40    |CØNTINUE
30    |CØNTINUE
20    |CØNTINUE
      |RETURN
      |END
      |TARJETAS DE CONTROL
      |DATOS
```

```
      STOP
      END
      SUBRØUTINE LEE (A, NFA, NCA)
      DIMENSIØN A(7,7)
      READ (5,8) ((A(I , J), J = 1 , NCA) , I = 1 , NFA)
    8 FØRMAT (10F 8.3)
      RETURN
      END
      SUBRØUTINE MUMAT (A,B,C,NFA,NCA,NCB)
      DIMENSIØN A(7 , 7) , B(7 , 7), C(7 , 7 )
      DØ 20 I = 1 , NFA
      DØ 30 J = 1 , NCB
      C(I , J) = 0.0
      DØ 40 L = 1 , NCA
      C(I , J) = C(I , J) + A(I , L)* B(L , J)
   40 CØNTINUE
   30 CØNTINUE
   20 CØNTINUE
      RETURN
      END
      SUBRØUTINE ESCR (NCPS,A,NFA,NCA)
      DIMENSIØN A(10,10),I2(10)
      IF(NCA-NCPS)13,14,13
   14 N10=1
      GO TO 15
   13 N10=NCA/NCPS+1
   15 DO 20 K=1,N10
      NIN=K*NCPS+1-NCPS
      NF1=K*NCPS
      I1=NIN-1
      NT1=NCA-(N10-1)*NCPS
      IF(K-N10)10,11,10
   10 DO 21 J=1,NCPS
      I2(J)=I+J
   21 CØNTINUE
  173 FØRMAT(//10(10X,I3)//)
  176 FØRMAT(//3X,I3,4X,10F12.4)
      WRITE(6,173)(I2(J),J=1,NCPS)
      GO TO 16
   11 DO 22 J=1,NT1
      I2(J)=I1+J
   22 CØNTINUE
      WRITE(6,173)(I2(J),J=1,NT1)
      NFI=NCA
   16 DO 23 J=1,NFA
      WRITE(6,176)J,(A(J,L),L=NIN,NFI)
   23 CØNTINUE
   20 CØNTINUE
      RETURN
      END
```

TARJETAS DE CONTROL

DATOS