Ing. William Chauca Nolasco
William5988@hotmail.com

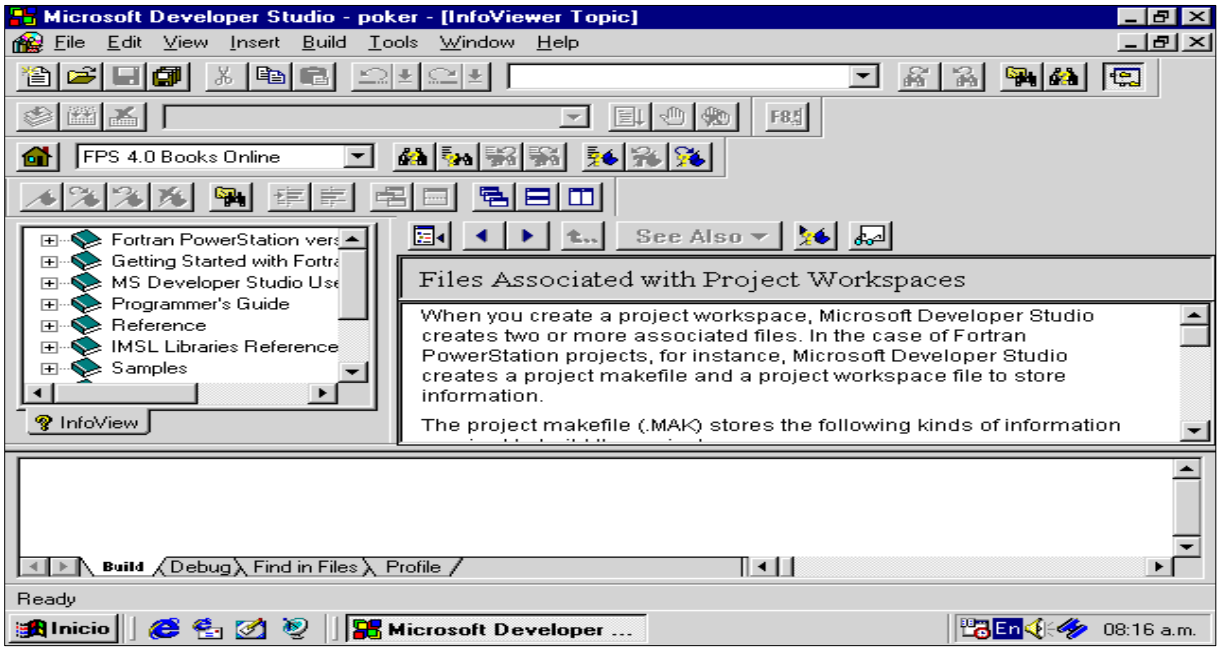
PROGRAMACION CON FORTRAN 90

Requerimientos del Sistema:

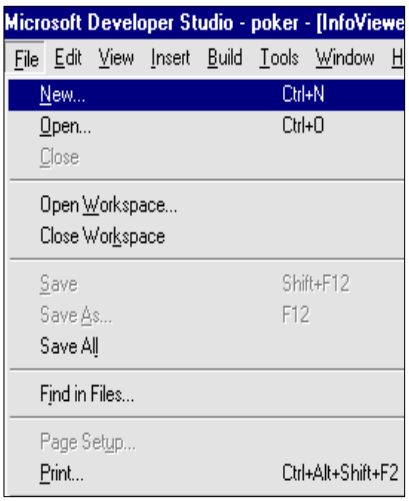
Para programar con Fortran 90, se necesitan:

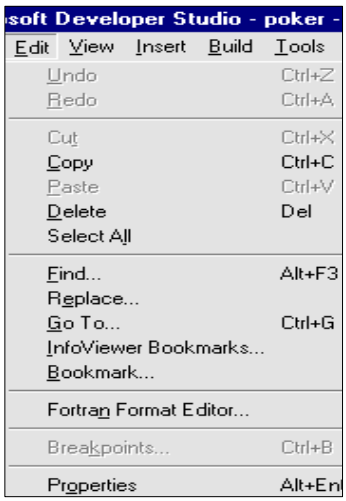
- Una Computadora Personal: 80386 - 80486 o Pentium, Memoria RAM 16 Mb (Mejor 32 Mb)
- El disco duro deberá tener un espacio suficiente para copiar los archivos, adicionalmente 45 Mb de espacio.
- Sistema Operativo Windows 95 - 98.
- Programa de instalación en CD - ROM del Fortran PowerStation
- Monitor a color SVGA, un Mouse.

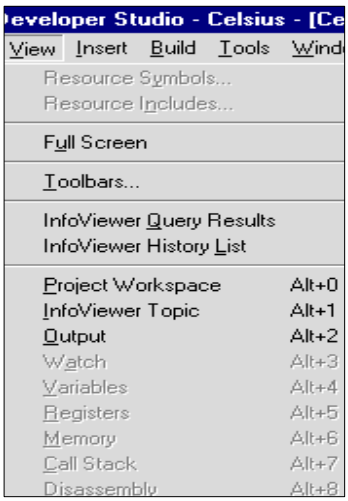
Entorno del Fortran 90 en Windows



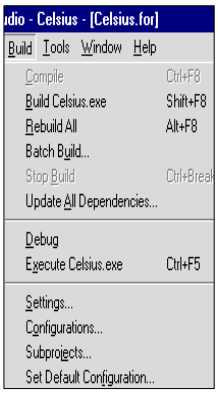
Opciones de Menú de la barra principal:

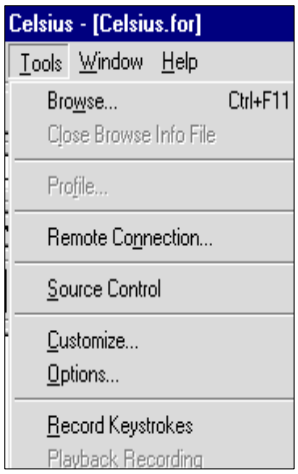


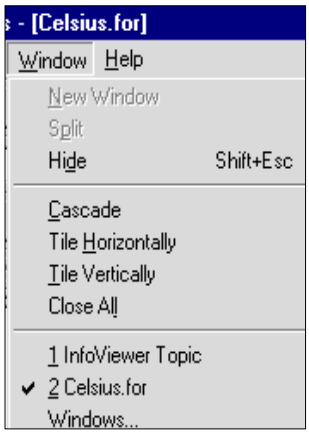


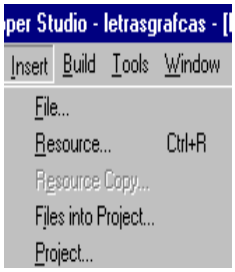


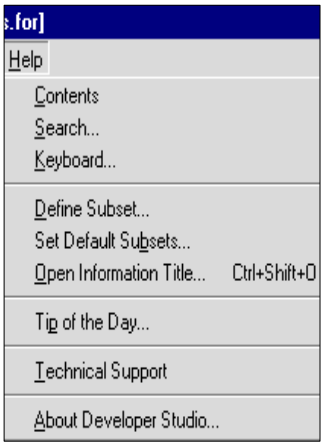
Ing. William Chauca Nolasco
96933582









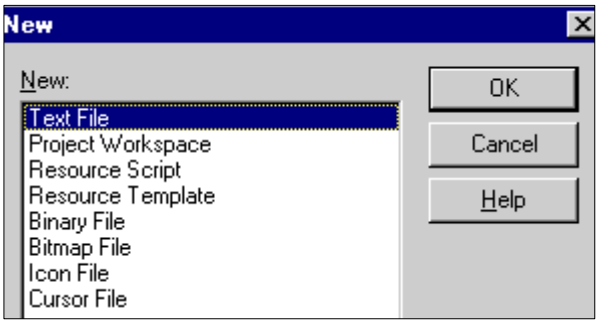


PASOS PARA CREAR UN PROGRAMA EN FORTRAN

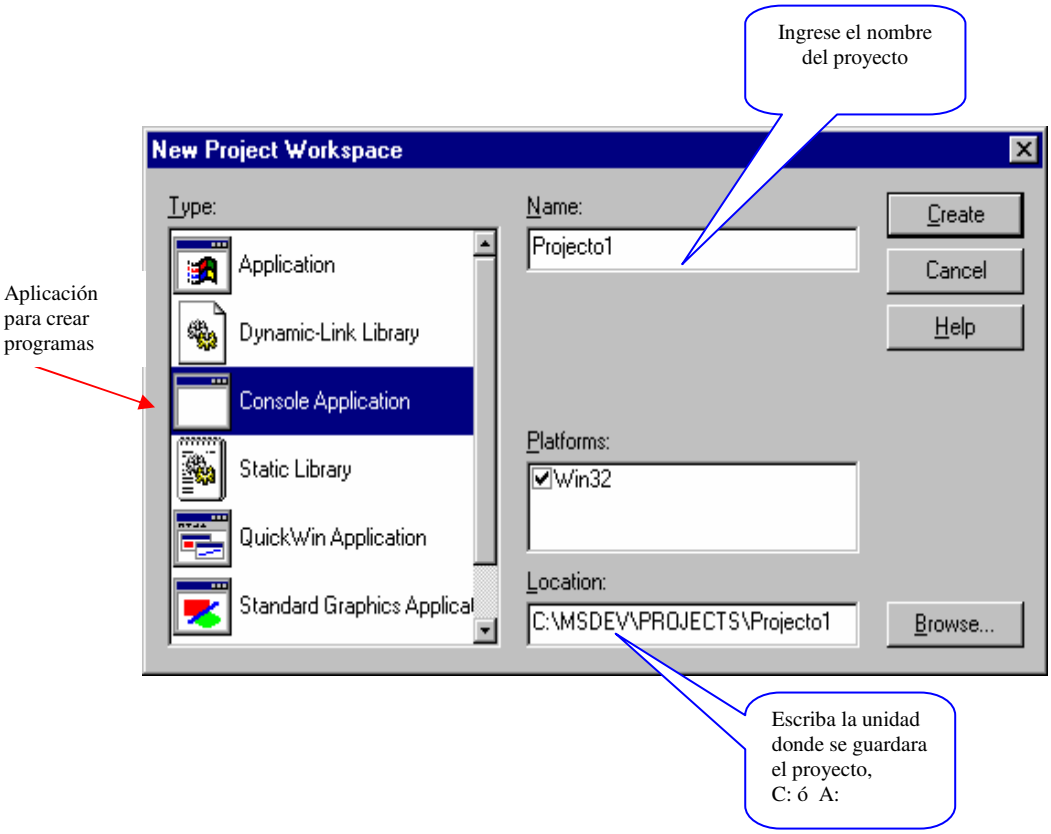
Primero se crea la carpeta de projectos, luego se genera el archivo de texto (.FOR) , para que sea compatible con las intrucciones del Fortran 77, o sino con el editor y grabarlo (.F90)

1. Ir al Menu File – New

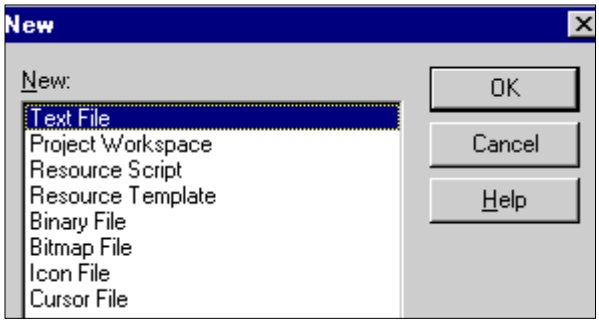
Seleccione la opcion Project Workspace, luego Ok.



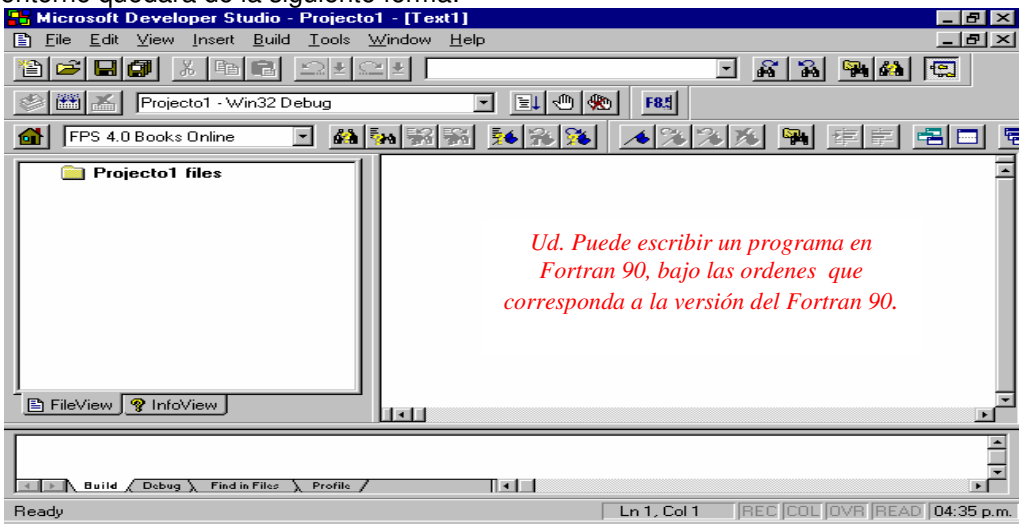
Sale una nueva ventana para crear el nuevo projecto, en esta ventana se debe especificar el nombre del proyecto y la unidad donde se deberá grabar el proyecto de tal forma que se almacenen todos los archivos que se crean a la hora de compilarlo, ejecutarlo el programa, es necesario que el archivo donde se encuentra el programa sea igual al nombre del proyecto. (Observe la ventana



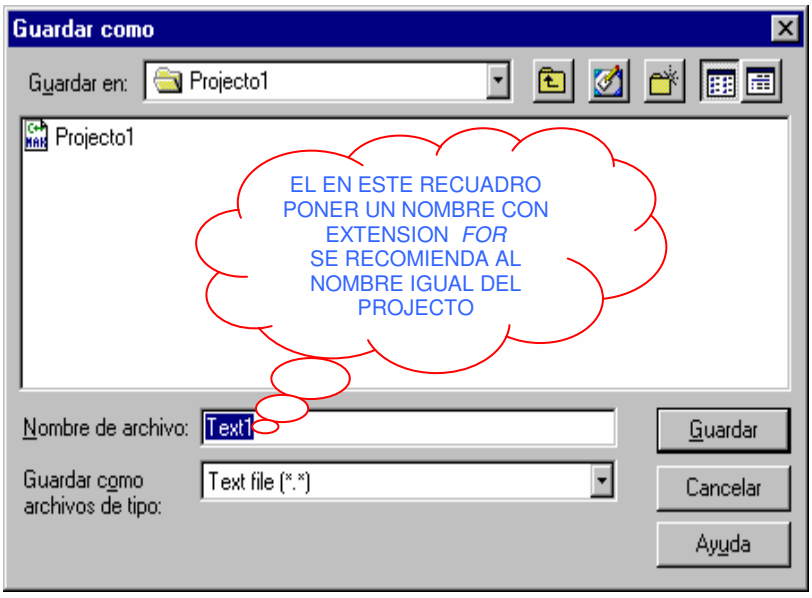
2. Nuevamente ir al Menu File – New, sale nuevamente el recuadro New, escoger la opcion Text File.



Luego el entorno quedara de la siguiente forma:



Si desea trabajar con instrucciones del Fortran para versiones anteriores hacer lo siguiente: Estando en la ventana anterior, ir al menu File – Save.



Ing. William Chauca Nolasco
96933582

El Lenguaje de Programación Fortran 90

Fortran 90 provee todos los aspectos originales del LENGUAJE FORTRAN 77, y agrega las extensiones y flexibilidad de nuevos idiomas. Algunos aspectos de la norma más antiguos se han declarado obsoleto, sin embargo el Fortran PowerStation reconoce los métodos más antiguos.

Fortran 90 agrega siete mejoras importantes respecto al LENGUAJE FORTRAN 77:

- Mejoras en lo numérico - capacidad de cómputo
- Orden en las operaciones
- Especificaciones de los datos intrísecos y precisión
- Declaración de datos definidos por el usuario
- Módulos para contener datos y los procedimientos usados por varias unidades de programa
- Indicadores
- Aprovisionamiento para la evolución de idioma

CAPITULO I

Introducción:

Un conjunto particular de reglas para codificar las instrucciones en una computadora se llama lenguaje de programación. Hay muchos lenguajes tales como por ejemplo Fortran, LENGUAJE BASIC, Pascal y C++. Fortran, es en español: TRADUCCION DE FORMULAS, es el primer lenguaje de programación de " alto nivel". Este lenguaje Hizo lo posible en usar nombres simbólicos para representar cantidades matemáticas, y para escribir fórmulas matemáticas en una forma razonablemente inteligible, tal como X = B/(2*A). La idea del Fortran se propuso 1953 por John Backus, en Nueva York, y el primer programa en Fortran ejecutado fué en Abril 1957.

Trabajando con programas simples en Fortran 90:

Si usted es nuevo a Fortran, deberá correr estos programas que son ejemplos que se muestran en el presente capitulo, sin tratar de comprender en forma detallada como trabajo. Las explicaciones seguirán en los capitulos siguientes. Usted ya sabe como crear un programa bajo el entorno del Fortran 90 en su computadora.

Ejemplo1. Este programa saluda si usted ingresa su nombre:

```
! Mi primer programa en Fortran 90
! Saludo
Character NOMBRE*20
Print*, “Cual es su nombre?”
Read*, NOMBRE
Print*, “Hola que tal:”, NOMBRE
End
```

Primero deberá compilarlo, ir al Menú BUILD – Compile
Segundo Construir el archivo ejecutable, Menú BUILD - Build
Tercero , Ejecutar el programa: Menú BUILD - Execute
Al ejcutar el programa: Saldrá en la pantalla de su monitor:

Cuál es su nombre?
Cecilia
Respuesta: **Hola que tal: Cecilia**

Ejemplo2. El programa siguiente computa el número de casos acumulados DE SIDA A(t) en los Estados Unidos en el año t según la fórmula: $A(t) = 174.6\ (t - 1981.2)^3$

```
PROGRAM SIDA
! Calcula número de casos acumulados DE SIDA en USA
Integer  T          ! año
Real     A          !  número de casos
Read*, T
A  = 174.6 * (T - 1981.2) ** 3
Print*, “Casos DE SIDA acumulados en USA para el año “, T, “:”, A
End Program SIDA
```

Usted ingresa el valor 2000 correspondiente a este año La respuesta es:
Casos DE SIDA acumulados en USA para el año 2000 : 1.1601688E+06
La respuesta se da en la notación científica. E+06 lo cual es el número 1.16 millónes.

Formato de un programa fuente en fortran:

- Es compatible con Fortran 77
- Incorpora un nuevo formato denominado Formato Libre (Free Form), donde las columnas no son significativas, pero los espacios en blancos sí.
- Una línea de código puede extenderse mas allá de la columna **72** (como máximo **132** columnas)
- Incorpora nuevos operadores, tales como: (==, /=, <=, >=, >, <)
- Se utiliza el punto y coma (**;**) para separar sentencias múltiples.
- Utiliza el simbolo **¡** para escribir comentarios en línea
- Utiliza el símbolo **&** como continuación de línea para una sentencia.

Resumen:

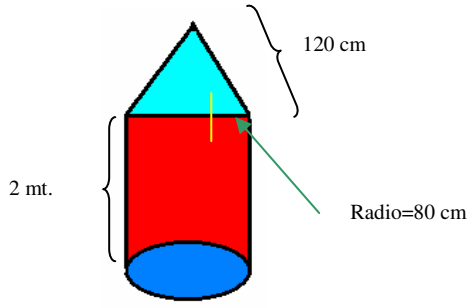
Como Ud. Ha observado en los dos ejemplos, un programa de computadora es un conjunto de instrucciones codificadas para resolver un problema particular.

- La sentencia de declaración **READ*** del Lenguaje Fortran se usa para ingresar datos en la computadora. (Free Form)
- La sentencia de declaración **PRINT*** del Lenguaje Fortran se usa para imprimir (mostrar) resultados. (Free Form)

EJERCICIOS - 1

1
2
3
4 $F = \left(\frac{9}{5}\right).C + 32$
5
6. yarda = 3 pies, 1pulgada = 2.54 cm, 1 m = 100 cm).

7. Unas camisas se venden en 10 dólares cada una si es que se compran más de tres y en 12 en otro caso. Escriba un programa en Fortran que sea un numero de entrada con la cantidad de camisas a adquirir e imprima el costo total.
8. Un trabajador recibe su sueldo normal por las primeras 30 horas y se le paga 1.5 vcces su sueldo normal por cada hora después de las primeras 30 horas.Escriba un programa que calcule e imprima el pago del empleado basado en el sueldo normal y el numero de horas trabajadas por el empleado, estos datos deberán ser introducidos por el usuario. Por ejemplo, si se indica al programa que las horas son 42 y 6.00 para el sueldo por hora, el programa debe imprimir 288.00 dólares como monto por percibir.
9. **Escriba un programa que permita la lectura de 6 números, calcule el promedio y imprima el promedio**
10. Elaborar un programa que calcule la superficie total del deposito de la figura, formado por un cono, un cilindro y una semiesfera.



11. La presión, el volumen y la temperatura de una masa de aire se relacionan por la formula:
    PV=0.37 m (T+460)
    Donde: P= presión en libras por pulgada cuadrada.
    V=volumen, en pies cúbicos.
    m=masa del aire, en libras.
    T=temperatura, en ºF
    Un neumático de automóvil tiene 2 pies cúbicos y se infla a una presión de 28 libras por pulgada cuadrada, a la temperatura ambiente. Elabore un programa que calcule cuanto aire hay en el neumático?
12. Supongamos que un automóvil parte del reposo y tiene una aceleración constante a por t segundos. La velocidad final v y la distancia d recorrida por el automóvil, son dadas por las formulas: $d = \frac{1}{2}at^2$ y v=at, escriba un programa que lea a y t y visualice t, d , v.

CAPITULO II

En este capítulo y en los proximos aprenderemos a escribir en forma detallada los programas en Fortran para resolver problemas simples. Hay dos requerimientos esenciales:

- Deben aprenderse las reglas exactas de las instrucciones para codificar el programa
- Un planeamiento lógico para resolver el problema a desarrollar. (Diagramas de flujo, pseudocodigos)

Estructura de un programa:

La estructura general de un programa simple en Fortran es como se indica a continuación (los artículos entre corchetes son opcionales):

[PROGRAM **nombre_del_programa**]
[declaraciones de sentencias]
[declaraciones ejecutables]
END [PROGRAM [**nombre_del_programa**]]

Como usted puede ver, la declaración obligatoria en un programa en Fortran es **END** esta declaración informa el compilador que no hay mas sentencias en Fortran compilar.

La sentencia **END** [**PROGRAM** [**nombre_del_programa**]] significa que el nombre de programa puede omitirse desde la declaración END, pero que si hay un nombre de programa, la sentencia **PROGRAM** no puede omitir dicho nombre.

Las declaraciones (sentencias) en un programa:

Las declaraciones forman la base de cualquier programa en Fortran, y puede contener desde 0 a 132 caracteres (una declaración puede ser vacía; las declaraciones vacías se realizan para hacer un programa más legible por secciones lógicas)

Todas las declaraciones del Fortran, exceptuando la declaración de asignación (p. ej. EQUILIBRIO = 1000), comienzan con una letra. Ejemplo **END, PRINT, PROGRAM**

Generalmente, una declaración debe escribirse en una línea. Sin embargo, las declaraciones múltiples pueden aparecer sobre una línea si ellos son separados por punto y coma (;). En aras de la claridad, estos se recomiendan unicamente con asignaciones cortas, tal como A = 1; B = 1; C = 1

Espacios en blanco:

Los espacios en blancos generalmente no son importantes, es decir usted los puede usar para mejorar y dar legibilidad a las declaraciones esta operación se le llama **sangrar**. Sin embargo, hay lugares donde blancos no son permitidos.

Ejemplos:

- INTE GER { observe que una declaracion en fortran no debe tener espacio en blanco}
- BALANCE AND < = {habría un error dado que **<=** es un operador}
- A * B se permite y es igual que A*B
- REALX {Esta declaracion de variable no esta permitido, debe tener un espacio en blanco}
- 30CONTINUE no son permitidos (30 es la etiqueta, debe tener un espacio en blanco)

Comentarios en un programa:

Si se escribe un juego de caracteres que siguen a una marca de exclamación (**!**), (No será un comentario si este simbolo esta dentro de una cadena de caracteres encerradas entre apostrofes o doble comillas) se denomina **COMENTARIO**, y son ignorados por el compilador. Una línea entera puede ser un comentario. Una línea vacía se interpreta también como comentario. Los comentarios deberían usarse para mejorar la legibilidad de un programa.

Continuación de lineas:

Si una declaración de programa es demasiado larga sobre una línea, y se desea continuar sobre la próxima línea debera colocar un simbolo tal como ampersand (**&**)

Ejemplo:

```
A = 174.6 *           &
   (T - 1981.2) ** 3
```

Un & al final de una línea de comentario no continuará el comentario, desde el simbolo & se interpreta como parte del comentario.

Tipos de Datos:

El concepto de tipos de datos es fundamental en Fortran 90. Estos consisten de un conjunto de valores de datos (por ejemplo los números enteros), una forma de denotar estos valores (por ejemplo: - 2, 0, 999), y se puede realizar un conjunto de operaciones (por ejemplo: aritméticos) que se permite sobre ellos.

El Fortran 90 contiene cinco tipos de datos intrínsecos (es decir incorporado) que se dividen en dos clases:

(a) Los tipos numéricos son los: enteros (Integer), real (Real) y complejo (Complex).
(b) Los no - numéricos que son el carácter (Character) y lógico (Logical).

Se debe tener en cuenta que existen tipos de datos derivados a partir de los datos intrinsecos.

Constantes literales:

Las constantes literales (llamado constante simplemente) son los símbolos usados para denotar los valores de un tipo particular, es decir los caracteres reales que pueden usarse. Antes de considerar una constante debemos estudiar en forma detallada como la información se representa en una computadora.

Bits ‘n Bytes:

La unidad básica de información en una computadora es un **bit**: lo cual representa dos únicos estados posibles, comúnmente describe como encendido y apagado. Los dígitos binarios 1 y 0 se usan para representar estos dos estados matemáticamente. La palabra "bit" en una contracción de "dígito binario".

Los números en la memoria de una computadora se representan con código binario, donde cada **bit** es una sucesión de potencia de 2. Los números decimales 0 a 15, por ejemplo, se codifican en binarios como se indica a continuación:

Tabla - 1

Decimal	Binario	Hexadecimal	Decimal	Binario	Hexadecimal
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Un **byte** (octeto) es la cantidad almacenada en la memoria de la computadora para representar carácter, y son ocho **bits** lo cual cada uno de ellos puede estar en dos que estados posibles, y puede ser 2^8 es decir 256 combinaciones diferentes.

La notación Hexadecimal (ver la tabla 1) se usa frecuentemente porque es más simple que el codigo binario. Cada dígito hexadecimal estan bajo la potencia 16.
.

Un **byte** puede ser representado por dos dígitos hexadecimal

Constantes enteros:

Las constantes literales enteros se usan para denotar los valores de tipo intrínseco entero. La más simple y la mayoría de la representación se escribe con un signo o sin signo (número entero).

Ejemplos:

- 1000
- 0
- +753
- 999999
- 2501

En el caso de una constante entera positiva, el signo + es opcional.

El rango de los enteros no especificado bajo la norma del fortran, entan dentro de: -32768 a +32767

El rango puede especificarse sobre una computadora en particular parar usar un parámetro grande, que amplie el espectro de los enteros intrinsecos (Kind parameter)

Los números enteros positivos pueden también ser representados en codigo binarios, o hexadecimal.

Ejemplos:

Binario : B'1011'
Octal : O0767'
Hexadecimal : Z'12EF'

Constantes Reales:

Estos datos se usan para denotar valores reales de tipo intrínseco , y puede tomar dos formas:

La primera forma es la que se llama forma posicional o punto fijo y consiste de una cadena de dígitos con un punto decimal. Pudiendo tener signo o sin signo. ejemplos:

- 0.09
- 37.
- 37.0
- .0
- - .6829135

No puede haber dígitos a la izquierda del punto decimal, o también dígitos a la derecha del punto decimal, pero un punto decimal por sí mismo no es permitido.

La segunda forma es la exponencial llamado también punto flotante, esta notación consiste básicamente de uan parte entera (con o sin signo) o un punto fijo real (con o sin signo) seguido en ambos casos por la letra **E** seguido por un entero (con o sin signo). El número que sigue a **E** es el exponente y nos indica la potencia de 10 por que el número que precede a **E** debe multiplicarse.
Ejemplos:

- 2.0E2 { = 200.0 }
- 2E2 { = 200.0 }
- 4.12E+2 { = 412.0 }
- - 7.321E-4 { = - 0.0007321 }

Las constantes reales se almacenan en forma exponencial por lo que los valores reales se almacenan de manera diferente a la de una constante de tipo entero. Por ejemplo. 43 es un entero, mientras 43.0 es real y estos se representarán de manera diferente en la memoria.

Nombres de Variables:

Reglas:

- El primer carácter del nombre de una variable debe ser una letra del alfabeto.
- Los siguientes caracteres pueden ser letras, digitos, se permite el caracter del subrayado (_)
- Los nombres de variables pueden tener de 1 hasta un máximo de 31 caracteres significativos. (alfanuméricos)
- Fortran 90 no hace distinción entre las letras mayúsculas o minúsculas. Ejemplo: NOMBRE es lo mismo que NomBre
- Generalmente un nombre de variable debe ser significativo: ejemplo: Numero_Estudiantes (Significativo), no escribir N (que no indica nada)
- No se debe utilizar las palabras reservadas del Fortran 90, tales como END, PROGRAM, INTEGER, para indicar un nombre de variables

La siguiente tabla muestra algunos nombres de variables válidos e inválidos.

Tabla 2

Nombre de variables válidos	Nombres de Variable no válidos
X	X+Y
R2R2	SUMA_DOS
CAL_VOLUMEN	2AB
FINDEMES	MULTI-TABLA

Un nombre de variable se ubica tomando una ubicación de memoria cuyo valor puede cambiarse durante la ejecución de un programa. El nombre de un variable se construye según las reglas dadas anteriormente. Un variable esta asociado a un tipo de dato que va a almacenar. Y Se da de acuerdo al tipo de declaración. Ejemplo:

INTEGER X ¡ Variable X
REAL INTERES ¡ Variable INTERES
CHARACTER LETRA ¡ Variable LETRA
REAL **::** A = 1

Notese que una variable puede ser inicializada en su declaración. En este caso debe usarse (**::**). El valor de un variable inicializada de esta manera puede cambiarse luego en el programa. Aunque las variables X, INTERES y LETRA se han declarado en el fragmento del programa arriba, ellos son hasta ahora indefinidos, dado que ellos no tienen ningún valor.

Una variable puede también ser declarado dando un valor inicial utilizando una sentencia DATA, Ejemplo:

```
REAL A, B
INTEGER I, J
DATA  A, B / 1, 2 / I, J / 0, - 1/
```

Un nombre (identificador después de la declaración PROGRAM) en un programa debe ser único. Por ejemplo, si un programa se nombra MONEY, un intento de declarar una variable del mismo nombre ocasionará un error.

Las variables descritas aquí son de forma escalar porque ellos pueden retener únicamente un valor.

Regla para los tipos de datos implicito:

Las versiones anteriores de Fortran90 tuvieron una regla de tipo implícita.

Las variables que comienzan con las letras **I** a **N** inclusive se especifica automáticamente como de tipo entero, mientras que las variables que comienzan con cualquier otra letra se especifican automáticamente como reales (**A** a **H**, **O** a **Z**). Esta regla todavía se aplica en Fortran 90, para asegurar la compatibilidad del código escrito eno versiones anteriores.

La regla de tipo implícita puede conducir errores a los programadores. Un valor real podría inconcientemente ser asignado a una variable de tipo entero entero (según las versiones anteriores); donde la parte de la fracción se truncaría, por ejemplo, la declaración.

Interes_tasa = 0.12

En un programa donde la variable **Interes_tasa** no es declarado explícitamente, asignará el valor 0 a la variable.
Para proteger contra tales errores se recomienda usar frecuentemente la declaración ***IMPLICIT NONE***

Esta declaración de debe usar al comienzo de todos los programas. Esta declaración cambia la regla de tipo implícita; consiguientemente todos los variables usados en el programa deben declararse.

Ejemplo3.- Un programa en Fortran 90, para calcular El Movimiento Vertical bajo la acción de la Gravedad

Si una piedra se tira verticalmente ascendente con una velocidad inicial ***u***, su desplazamiento vertical ***s*** después de un tiempo ***t*** transcurrido esta dado por la fórmula : ½ Vt – gt^2 /2, donde g es la aceleración debido a la gravedad. La resistencia de aire ignora. Si quisiéramos calcular el valor de ***s***, dando ***u*** y ***t***.

La preparación lógica de este programa es como se indica a continuación:

1. Ingresar valores de ***g***, ***u*** y ***t*** en la computadora
2. Calcular el valor de ***s*** según la fórmula
3. Imprimir el valor de ***s***
4. Parar

Este plan puede parecer trivial para su persona, y un derroche de tiempo escribiendo.

Aún usted se sorprendería cuántos principiantes, prefieren que se realice directamente a la computadora, trate de programar el paso 2 antes del paso 1. Como observa Ud., se debe tener una disciplina, primero se debe escribir en el papel los pasos a realizar para obtener el resultado sastifactorio (Psedocodigos / Diagramas de Flujos)

```
PROGRAM Vertical
! El movimiento vertical bajo la accción de la gravedad
IMPLICIT NONE
REAL, PARAMETRO :: G = 9.8 !  aceleración debido a la gravedad
REAL S          ! desplazamiento (metros)
REAL T          !  tiempo
REAL U          ! velocidad inicial (metros/seg)
PRINT*, " Tiempo   Desplazamiento"'
PRINT*
U = 60
T = 6
S = U * T - G / 2 * T ** 2
PRINT*, T, S
END PROGRAM Vertical
```

La Tabla 3 presentan los operadores numéricos intrínsecos

Tabla 3

Operador	Precedencia	Significado	Ejemplo
**	1	Exponenciación	2**4 = 16
*	2	Multiplicación	2 * A
/	3	División	B / DELTA
+	4	Adición	A + 5.6
-	4	Sustracción	B – C

Nota: Se debe tener en cuenta que para expresiones que se escriben con más de un operador se deben encerrar entre paréntesis ( ).

Expresiones numericas:

El programa escrito en el ejemplo 3, hace uso del código siguiente:
U * T - G / 2 * T ** 2. Este es un ejemplo de una expresion numérica lo cual es una combinación de constantes, variables (además puede contener funciones intrínsecas como la raíz cuadrada) usando operadores intrínsecos numéricos. Se debe tener en cuenta una regla para computar un valor.

Un operador con dos operandos, como en ***A + B***, se llama binario o el operador ***dyadic***r.

Cuando un operador aparece con un único operando, como ***- Z***, se llama unario o ***monadic.***

La orden en que que se realizan operaciones en una expresión mas compleja se debe tener en cuenta la precedencia de los operadores, según la **Tabla 3**, de existir parentesis ( ) siempre tendrá la precedencia más alta.

Ejemplo la multiplicación tiene una precedencia más alta que la adición, esto significa, por ejemplo, que 1 + 2 * 3 se evalúa como 7, mientras (1 + 2) * 3 se evalúa como 9. Anote también que - 3 ** 2 evalúa a –9, no 9.

Cuando un operador con la misma precedencia se da en una misma expresión, la forma de evaluar será siempre de izquierda a derecha, así:
1 / 2 * A, se evalua como (1 / 2) * A y no 1 / (2 * A).

La excepción a esta regla de precedencia es que en una expresión de la forma: A ** B ** C la operación se realiza de derecha a izquierda B ** C se evalúa primero.

División de ENTEROS:

Cuando una cantidad de tipo entero (constante, variable o la expresión) es dividida por otro entero el resultado es también de tipo entero, Se debe tener en cuenta que se trunca hacia cero, es decir la parte de la fracción se pierde. Ejemplos:.

- 10 / 3 Da como resultado 3
- 19 / 4 Da como resultado 4
- 4 / 5 Da como resultado 0
- - 8 / 3 Da como resultado - 2
- 3 * 10 / 3 Da como resultado 10
- 10 / 3 * 3 Da como resultado a 9

Expresiones de modo mixto (reales y enteros):

Fortran 90 permite perandos en una expresión de tipos diferentes. La regla general es que el tipo más débil o más simple se convierte al tipo más fuerte. El tipo entero es el más simple, esto significa que las operaciones que involucran operandos reales y enteros se hará en la aritmética real. Ejemplos

- 10 / 3.0 Evalúa a 3.33333
- 4. / 5 Evalúa a 0.8
- 2 ** (- 2) Evalúa a 0 , por qué?

Sin embargo:
- 3 / 2 / 3.0 Evalúa a 0.333333 porque 3/2 se evaluara primero, dando el entero 1.

Asignación Numérica:

El propósito de calcualr una expresión numerica es obtener dicho valor y asignarlo a una variable. Su forma general es: ***Variable = Expr***

El signo igual no tiene el mismo significado como el signo igual en la matemáticas, y debera leerse como **"Asignar a "**. La asignación:

X = A + B

Debe leerse como "(El contenido de) X llega a ser (los contenidos de) A más (los contenidos de) B".

De esta manera la asignación:

N = N + 1, Es significativo, y los medios "aumentan el valor de N por 1", considerando la ecuación matemática

Si ***expr*** no es del mismo tipo como ***var***, este se convierte a el tipo antes de la asignación. Esto significa que puede haber pérdida de precisión.

Por ejemplo, suponiendo ***N*** es entero, ***X*** y ***Y*** son reales:

- N = 10. / 3 (el valor de N es 3)
- X = 10 / 3 (el valor de X es 3.0)
- Y = 10 / 3. (el valor de Y es 3.33333)

El peligro de las divisiones de enteros no pueden acentuarse demasiado. Por ejemplo, usted podría querer promediar dos de marcas que suceden para ser enteros M1 y M2. La declaración más natural para escribir es:

FINAL = (M1 + M2) / 2

Pero se pierde la parte decimal del promedio. Es siempre más seguro escribir la constante como real si la aritmética real es qué usted quiere:

FINAL = (M1 + M2) / 2.0

Ejemplos:

La fórmulas se traduce en el Fortran:

```
F = G * M * E / R ** 2
C = (A ** 2 + B ** 2) ** 0.5 / (2 * A)
A  = P * (1 + R / 100) ** N
```

El segundo ejemplo se puede escribir con ***SQRT*** función intrínseca como

C = SQRT (A ** 2 + B ** 2) / (2 * A), pero nunca como:
C = (A ** 2 + B ** 2) ** (1/2) / (2 * A), (1/2 es el exponente y se evalúa a cero a causa de la división entera).

Sentencias simple de Entrada y Salida de datos:

En esta sección nosotros estudiaremos las declaraciones READ* y PRINT* más estrechamente. El proceso de ingresar información y obtener fuera de la computadora es un aspecto se llama ***Transferencia de datos***.

Entrada (lectura) de datos (Input)

Si dentro de un programa se declaran estas sentencias con valores asignados a estas variables por ejemplo:

BALANCE = 1000
TASA = 0.09

Esta es una manera inflexible de surtir datos, sin embargo al ejecutar el programa si se quisiera que las variables BALANCE y TASA se ejecutan para diferentes valores debera asignarle diversos valores repitiendo en el programa varias veces. La sentencia o declaración READ* sin embargo, permite abastecer los datos mientras el programa corre o se ejecuta. Podemos reemplazar estas dos declaraciones de asignación con una única declaración.

READ*, BALANCE, TASA

La forma general de la Sentecia ***READ**** es: ***READ****, ***lista***

Donde: ***lista*** es una lista de variables separada por comas.

Las reglas generales para la declaración *READ**

- Una simple línea de entrada o salida se llama un ***registro*** (por ejemplo en el caso de una PC, desde el teclado o la pantalla).

- Cada declaración READ requiere un nuevo registro de aporte. Por ejemplo la declaración:

READ*, A, B, C

Se estará satisfecho con un registro que contiene tres valores:

3 4 5 {Cada valor se escribe pulsando la barra espaciadora)
Considerando las declaraciones:

READ*, A
READ*, B
READ*, C

Requiere tres registros de aporte, cada uno con un valor:

3 (Escribir el valor 3 por teclado y pulsar ***ENTER***)
4 (Escribir el valor 4 por teclado y pulsar ***ENTER***)
5 (Escribir el valor 5 por teclado y pulsar ***ENTER***)

- Si no hay los datos suficientes para satisfacer una sentencia READ el programa visualizarán un mensaje de error.

Ejemplo: Las declaraciones
READ*, A
READ*, B, C
READ*, D

Con el aporte de los siguientes registros
1 2 3
4
7 8
9 10

Teniendo el siguiente efecto con las asignaciones:
A = 1
B = 4
C = 7
D = 9

Lectura de de datos desde archivo de texto:

Frecuentemente sucede que usted necesita probar un programa leyendo muchos datos. Suponga que se escribe un programa para encontrar el promedio de 10 números.

La idea está en poner los datos en un archivo externo que se almacenan sobre su sistema de computadora, por ejemplo, su disco duro.

El programa entonces lee los datos desde el archivo cada vez que se ejecuta, en vez de que Ud. Ingrese por teclado dichos numeros.

Como ejemplo, se puede usar un procesador de textos para almacenar los datos en archivo de tipo ASCII (texto) (puede usar el editor de Worpad, Notepad, MsWord, y grabarlo en formato Texto). El archivo puede tener el nombre como **DATOS**

3 4 5

Ahora usaremos un programa para leer estos tres números desde el archivo **DATOS** y luego mostrarlo sobre la pantalla:

```
OPEN( 1, FILE =' DATOS' )
READ (1,*) A, B, C
PRINT*, A, B, C
END
```

La sentencia ***OPEN*** conecta a la unidad (dispositivo) es el número (**1**) al archivo externo ***DATOS***. La forma de la sentencia ***READ*** mostrada en el ejemplo dirige al compilador para mirar el archivo conectado a la unidad 1 para leer sus datos (el número de unidad puede típicamente estar desde 1 a 99).

Salida de datos (Output)

La sentencia ***PRINT**** es muy útil para mostrar cantidades pequeñas de datos, comúnmente mientras usted desarrolla un programa, no necesita tener preocupación con los detalles exactos de la forma de obtener los resultados.

El formato general es: ***PRINT*, lista***

Donde:
Lista : puede ser una lista de constantes, variables, expresiones, y cadena de caracteres, separa por comas. Una cadena de carácteres es una sucesión de caracteres delimitada por comillas (") o apóstrofes ('). Por ejemplo:

PRINT*, "La raíz cuadrado de", 2, 'es', SQRT(2.0)

Aquí mostramos algunas reglas generales:

- Cada sentencia PRINT* genera un nuevo registro de rendimiento.

- La manera que se imprimen los numeros reales depende de su sistema (Pc) en particular. El compilador Fortran 90 en un procesador 386, por ejemplo, muestra los números reales entre: –99.999 y +99.999 en el formato de punto fijo, y los otros procesadores de más alto orden (486/586-Pentium) en forma exponencial. Si usted quiere más precisión de los datos de salida, aprenderemos mas adelante a usar formato de salida de datos. Por ejemplo, las declaraciones siguientes imprimirán el número 123.4567 en la forma fija de punto sobre 8 columnas corriendo a dos lugares decimales:

```
      X = 123.4567
      PRINT* 10, X
10    FORMAT( F8.2 )
```

- Si en una cadena de carácteres la declaración PRINT* se anhela escribir demasiado caracteres sobre una de línea se mostrará sin una rotura si usa **&**, también aparece en la línea de continuación:

Ejemplo:
PRINT*, “Ahora es el tiempo para que todos los **&**
& hombres muestren la resistencia contra la corrupción “

Enviar datos a la impresora

Esto puede hacerse como se indica a continuación (sobre una Computadora Personal):

OPEN(**2**, FILE = **'prn'**)
WRITE*(2, *) “Viva la Nueva Democracia?” ¡Sale por la Impresora
PRINT*, “Muera el Abuso de los Ursupadores” ¡Sale por la pantalla

NOTA:
La sentencia WRITE debe usarse conjuntamente con un número de unidad de dispositivo. Esta es una declaración más general que PRINT.

Ejercicios y Problemas propuestos:

1 Evaluar las expresiones numéricas siguientes, sabiendo que: A = 2, B = 3, C = 5 (Reales); y I = 2, J = 3 (Enteros). la Respuestas se dan en los parentesis.

- A * B + C (11.0)
- A * (B + C) (16.0)
- B / C * A (1.2)
- B / (C * A) (0.3)
- A / I / J (0.333333)
- I / J / A (0.0)
- A * B ** I / A ** J * 2 (4.5)
- C + (B / A) ** 3 / B * 2. (7.25)
- A ** B ** I (512.0)

- - B ** A ** C (- 45.0)
- J / (I / J) (división por cero)

2 Decidida cual de las constantes siguientes no son válidos en Fortran estándard, y confirme por qué no:

(a) 9,87 (b) .0 (c) 25.82 (d) –356231
(e) 3.57*E2 (f) 3.57E2.1 (g) 3.57E+2 (h) 3,57E–2

3 Diga ud. Porque no son nombres de variables en Fortran:

(a) A2 (b) A.2 (c) 2A (d) 'AONE
(e) AONE (f) X_1 (g) MiXedUp (h) Paga HOY
(i) U.S.S.R. (j) Pay_HOY (k) min*2 (l) PRINT

4 Encontrar los valores de las expresiones siguientes escribiendo programas cortos (respuestas esta en parentesis):

- La suma de 5 y 3 dividido por su producto (0.53333)
- La raíz cubica del producto de 2.3 y 4.5 (2.17928)
- El cuadrado de 2π (39.4784, para pi = 3.1415927)

5 Traducir las expresiones siguientes en Fortran:

5.1 $ax^2 + bx + c = 0$

5.2 $0 \le X \le x$

5.3 $\Phi(x) = 0.5 - r\left(at + bt^2 + ct^3\right)$

5.4 $r = \exp\left(-0.5x^2\right) \Big/ \sqrt{2\pi}$

5.5 $t = 1 \Big/ \left(1 + 0.3326x\right)$

6. Escribir un programa para calcular x, donde A = 2, B = - 10, C = 12 (Use la sentencia Read* para ingresar los datos). (Rpta. 3.0)

$$X = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

7. Escribir un programa en Fortran que intercambien los contenidos de dos variables A y B, usando una variable auxiliar T.

8. Reconocer los errores de sintaxis en este programa antes de compilarlo y ejecutarlo en la computadora.

```
PROGRAM Error -Full
REAL: A, B, X
X:= 5
Y = 6,67
B = X \ Y
PRINT* 'La respuesta es",  B
END.
```

CAPITULO III

En en el capitulo anterior se ha aprendido, como leer datos ejecutando un programa, realizar operaciones aritmeticas dentro de ellos, así como obtener resultados a traves de la pantalla de su monitor.

Ahora, aprenderemos a utilizar dos sentencias muy poderosas para construir programas más reales y complejos tales como la sentencias: **DO** y **IF**.

También aprenderemos, el uso de otros dos tipos de datos intrinsecos como: Tipo **Character** y el Tipo **Complex**. Mostraremos el uso de algunas funciones de biblioteca del Fortran.

Sentencia de Repetición: DO (Loops)

Esta sentencia lo aplicaremos con el ejemplo (1) siguiente:

```
Program Uso_DO
Integer I
Real R
DO I = 1,10
Print*, I
END DO
End Program Uso_Do
```

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
Press Any Key  to Continue
```

La ejecución de este programa, visualizará en la pantalla los valores de 1 hasta 10, linea por linea. Pero como se ha declarado una variable R (real), a la hora de compilarlo nos dará un mensaje de Cuidado (Warning) por el no uso de esta variable en el programa.

Sin embargo, si queremos generar numeros aleatorios 10 veces, dentro del bucle de repeticion podemos modificar el programa anterior.

```
PROGRAM Uso_DO
INTEGER I
REAL R
  DO I = 1,10
  CALL RANDOM_NUMBER(R)
  PRINT*,I ,R
  END DO
END PROGRAM Uso_Do
```

```
 1   2.247794E-05
 2   8.503245E-02
 3   6.013526E-01
 4   8.916113E-01
 5   9.679557E-01
 6   1.896898E-01
 7   5.149758E-01
 8   3.980084E-01
 9   2.629062E-01
 10   7.435125E-01
Press any key to continue
```

Si se quiere que el proceso de repetición de un bucle de sentencias se realizen en el orden inverso, es decir que la salida sea de 10 hasta 1, se puede modificar la sentencia Do, de la siguiente forma: DO I = 10,1,-1. (para el primer ejemplo):
Luego la salida de datos en la pantalla será:

```
 10
 9
 8
 7
 6
 5
 4
 3
 2
 1
Press any key to continue
```

La sentencia **DO (estructurado)** es una de la más poderosa en el Lenguaje Fortran, uno de sus formato más simple es:

DO I = J, K
[*Bloque de sentencias*]
END DO
Donde:

I es una variable entera, **J** y **K** son también expresiones enteras, y [Bloque de sentencias], representa cualquier número de declaraciones que se van a repetir.

En la primera vuelta " **I** " toma el valor de **J** , y entonces en la segunda vuelta se adiciona en 1 siempre al final de cada vuelta (incluso en la última vuelta), y para salir del lazo de repetición (loop) el valor de **K,** debe estar en **K+1** en la última vuelta.

Ejemplo 2. Podemos obtener la raiz cuadrada de cualquier número X usando operaciones aritméticas tales como la suma, resta y división, basado en el método de Newton. (Proceso iterativo). Para ello realizaremos un algoritmo de programa para obtener la raiz cuadrada para A = 2.:

- Ingresar un valor para → A
- Inicializar la variable de X a 1
- Hacer que se repita 6 (seis) veces las sentencias

  Reemplace X por $\frac{\left(X + \frac{A}{X}\right)}{2}$

  Imprima el valor de X
- Fin de la repetición.

```
Program Metodo_Newton
¡Obtener la raiz cuadrada con el método de Newton
IMPLICIT NONE
REAL A
INTEGER I
REAL X
WRITE( *, 10, ADVANCE = 'NO') "Ingrese numero:"
10  FORMAT (A)
 READ*, A
 PRINT*
 X = 1       !Inicializamos la variable x a 1
    DO I = 1,6
      X = (X + A / X) / 2
      PRINT*, X
    END DO  ¡Esta sentencia END DO es equivalente a ENDDO
PRINT*
PRINT*, "Uso de SQRT:", SQRT(A)
END PROGRAM Metodo_Newton
```

La salida en la pantalla es una secuencia de valores en la cual debe converger al valor deseado y debe ser idéntico a la evaluación por la función intrinséca: SQRT(a)

```
Ingrese numero 2

     1.500000
     1.416667
     1.414216
     1.414214
     1.414214
     1.414214
Uso de SQRT:       1.414214
Press any key to continue
```

Uso de la Sentencia de Desición Doble: IF--THEN – ELSE:

Para hacer uso de esta sentencia, lo haremos con un ejemplo práctico, cuyo enunciado es el siguiente:

Se tiene un archivo externo de datos de tres alumnos que incluye las notas de un curso cuyo puntuación varía de 0 a 100 , tal como se ve en:

Nota 1 Nota 2 Nota 3
40 60 43
60 45 43
13 98 47

Este archivo se debe crear con formato de tipo texto dentro del Fortran 90, el nombre del archivo se llama NOTAS.F90, lo cual deberá ser leído por el programa, deberá visualizarse las notas de los examenes, el promedio, así como un mensaje si paso o no paso el curso para aqueLlos que obtuvierón un promedio de puntuación mayor a 50.

```
PROGRAM MENSAJE_FINAL
IMPLICIT NONE
REAL PROME              !Promedio de las dos notas
REAL FINAL              !Variable auxiliar
REAL P1                 !Nota primer examen
REAL P2                 !Nota segundo examen
REAL P3                 !Nota del tercer examen
INTEGER STU             !Variable contador para el uso del Do
OPEN(1, FILE = 'C:\MIS DOCUMENTOS\NOTAS.F90')
PRINT*, "       Nota1     Nota2     Nota3  PROMEDIO OBSERVACION"
PRINT*
DO STU= 1,3
  READ(1, *) P1, P2, P3
  PROME = (P1+P2+P3)/3.0
      IF (PROME > 50.0) THEN
          FINAL = PROME
      ELSE
          FINAL = PROME
     END IF
       IF (FINAL >= 50) THEN
           PRINT*, P1,P2,P3,FINAL," PASO"
       ELSE
          PRINT*, P1,P2,P3,FINAL," NO PASO"
       END IF
END DO
END PROGRAM MENSAJE_FINAL
```

Nota1	Nota2	Nota3	PROMEDIO	OBSERVACION
40.000000	60.000000	43.000000	47.666670	NO PASO
60.000000	45.000000	43.000000	49.333330	NO PASO
13.000000	98.000000	47.000000	52.666670	PASO

Press any key to continue

La sentencia if-then-else, evalua una condición, si es verdadera, ejecuta el bloque que sigue a THEN, y si es falso ejecuta el bloque que sigue a ELSE.

Sintáxis:

IF <condición>
THEN
[bloque de instrucciones]
ELSE
[bloque de instrucciones]

END IF

Nota: en la condición viene a ser una expresión logica, pudiendo ser verdad o falso, estas expresiones contienen operadores relacionales, tal como <, <=, >=, /=, ==, además de los operadores logicos tales como: .NOT., .AND., .OR.

VARIANTES DEL IF:

El if lógico (declaración corta)
Sintáxis:

IF (Expresión1 <condición> Expresión2) *Instrucción*

Si la condición es verdadera se ejecuta la única instrucción que sigue a la condición.
Si la condición es falsa no se ejecuta esa instrucción.

Operadores relacionales:

Operador relacional	**Descripción**	**Ejemplo**
.LT. o <	menor que	A < 1e-5
.LE. o <=	menor o igual	B ** 2 .LE. 4 * A * C
.EQ. o ==	igual	B ** 2 == 4 * A * C
.NE. o /=	no igual	A /= 0
.GT. o >	mayor que	B ** 2 - 4 * A * C > 0
.GE. o >=	mayor igual que	X >= 0

Operadores lógicos:

Fortran 90 tiene cinco operadores lógicos, que operan sobre expresiones lógicas:

Operador lógico	Precedencia	Significando
.NOT.	1	Negación lógica
.AND.	2	Y lógico
.OR.	3	O lógico
.EQV.	4	Equivalente
.NEQV	4	No - equivalente

La siguiente " Tabla de la verdad" muestra los efectos de estos operadores sobre las expresiones lógicas LEX1 y LEX2 (T = cierto; F = falso):

LEX1	LEX2	**.NOT.** LEX1	LEX1 **.AND.** LEX2	LEX1 **.OR.** LEX2	LEX1 **.EQV**. LEX2	LEX1 **.NEQV.** LEX2
T	T	F	T	T	T	F
T	F	F	F	T	F	T
F	T	T	F	T	F	T
F	F	T	F	F	T	F

La orden de recedencia, mostrada en la tabla anterior, puede modificarse de existir parentesis, la cual siempre tiene la precedencia más alta.

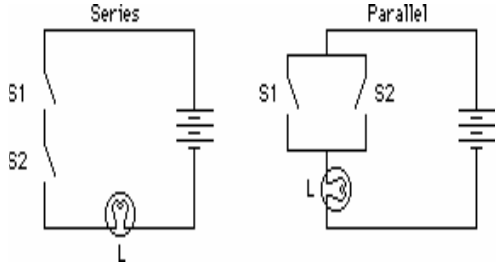
Ejemplos:

- (B * B == 4 * A * C) .AND. (A /= 0)
- (Final >= 60) .AND. (Final < 70)
- (A /= 0) .or. (B /= 0) .or. (C /= 0)
- .not. ((A /= 0) .and. (B == 0) .and. (C == 0))

Variables lógicas:

Una variable puede declararse de tipo lógico con la sentencia *LOGICAL*.. Las expresiones o constantes lógicos pueden asignarse a variables lógicos:

Esquema de Circuitos - Conmutadores:




```
LOGICAL L1, L2, L3, L4, L5
REAL A, B, C
...
L1 = .TRUE.
L2 = B * B - 4 * A * C >= 0
L3 = A == 0
L4 = L1 .and. .not. L2 .or. L3
L5 = (L1 .and. (.not. L2)) .or. L3
```

(Las reglas de precedencia hacen L4 y L5 sean lógicamente equivalente.)

Los valores de verdad de las variables lógicas son representados por T y F en la lista - dirigido I/O.

Simulación para el circuito de conmutación:

En el siguiente segmento del programa los variables lógicas S1 y S2 representa el estado de dos interruptores (ON = Cierto-Verdadero-Encendido; OFF = Falso-Apagado) y L representa el estado de un foco de luz. El programa muestra los circuitos en la figura donde los interruptores se arreglan o serie y el otro en paralelo.

```
LOGICAL L, S1, S2
READ*, S1, S2
L = S1 .and. S2    ! serie
L = S1 .or. S2     ! paralelo
PRINT*, L
```

Cuando los interruptores están en serie, la luz estará encendido únicamente si ambos interruptores están conectados. Esta situación es representada por S1.AND.S2

Cuando los interruptores están en paralelo, la luz estará encendido si uno o ambos de los interruptores están conectados. Este es representado por S1.OR.S2

LA SENTENCIA CASE (Selección Múltiple)

La sentencia CASE es similar a la sentencia IF. CASE permite seleccionar una situación entre un número de situaciones o casos, en base a un selector. En tales casos es más conveniente usar esta sentencia que el IF.

Consideremos el segmento del siguiente de programa:

```
CHARACTER CH
DO
READ*, CH
PRINT*, ICHAR( CH )
IF (CH == '@') EXIT
IF (CH >= 'A .and. CH <= 'Z' .or. CH >= 'a' .and. CH <= 'z') THEN
SELECT CASE (CH)
CASE ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u)
PRINT*, 'Vocal'
CASE DEFAULT
PRINT*, 'Consonante'
END SELECT
ELSE
PRINT*, 'Otra Cosa '
END IF
END DO
```

```
1
       49
OTRO CASO
A
       65
VOCAL
U
       85
VOCAL
I
       73
VOCAL
@
       64
Press any key to continue
```

Este programa a la hora que se ejecuta, Ud. Deberá presionar un tecla, lo cual decide si dicho tecla que corresponde a un carácter es una vocal, una consonante o otra cosa. Cuando presiona el simbolo @ el programa leerá y saldrá del programa dando el mensaje “Press a Key to continue” . También se puede usar la sentencia IF, pero seria un programa más extenso y engorroso.

SINTAXIS DE LA SENTENCIA CASE:

```
SELECT CASE (expr)
        CASE (selector1)
                [block1]
        CASE (selector2)
                [block2]
        [CASE DEFAULT
                 blockOTRO]
END SELECT
```

Nota:

Donde:

expr = Debe ser de tipo entero, caracter o lógico. Si evalúa el selector particular, entonces el bloque es ejecutado, de otra forma CASE DEFAULT se selecciona. CASE DEFAULT es

opcional, pero solamente es único. No necesariamente tiene que ser la última cláusula del CASE.

La forma general del selector es una lista de no - superposición de valores y rangos del mismo tipo como <expr>, encerrados entre parentesis, por ejemplo.

CASE('a:'h', 'í':'n', 'o':'z', '_')

Anote que los dos puntos puede usarse para especificar un rango de valores. Si el límite superior de un rango esta ausente, el CASE selecciona si <expr> evalúa al mayor valor del límite más inferior, y viceversa.

Las partes de la sentencia CASE pueden nombrarse del mismo modo como la sentencia IF.

Ejemplo:
Read(*,*) Idia
Select CASE(Idia)
Case(1)
    Write(*.*) “Lunes”
Case(2)
    Write(*.*) “Martes”
Case(3)
    Write(*.*) “Miércoles”
Case Default
    Write(*.*) “Otro día”
End Select

Resumen de la Sentencia Select Case:

1. Transfiere el control a un bloque de sentencias seleccionadas de acuerdo al valor de una expresión.
2. Es como una secuencia de Ifs Anidados.
3. Sintáxis:
   Select case (expresion)
   Case(lista1)
   ……Bloque de instrucciones1
   Case (lista2)
   ……Bloque de instrucciones2
   …….
   CASE DEFAULT
   ……Bloque de instruccionesN
   END SELECT

4. Lista representa:
   Valores separados por , 4,5,6
   Rangos delimitados por : 1:10, 20:23
5. Expresion debe resultar en un valor entero, logico o carácter(1)
6. El bloque del Case se puede repetir cuantas veces sea necesario.
7. La opcion Case Default es opcional.
8. Se evalúa la expresión
9. Si el valor coincide con un valor en lista1, se ejcuta el bloque de instrucciones1 y luego y luego se desvía después del END SELECT
10. Sino se verifica si el valor coincide con un valor en lista2 y se ejecuta el bloque de instrucciones2 y luego se desvia después del END SELECT.
11. Si el valor no coincide con ninguna lista se ejecuta:

- El bloque de comandosN (si existe)
- La instrucción siguiente al END SELECT si la opción Default no existe.

CARACTERES: (CHARACTER)

Una deficiencia al escribir un programa es que los nombres de los estudiantes no son leidos ni imprimidos. Para remediar esto nosotros hacemos uso de variables de tipo carácter. Ejemplo de lineas de declaraciones

CHARACTER (Len = 15) Nombre ! Nombre
Cambios de declaración que imprime el título:
PRINT*, 'Nombre Nota1 Nota2 Nota3 Promedio Observacion'

Utilizando la declaración READ

READ(1, *) Nombre, Nota1,Nota2,Nota3

Cambiando las dos declaraciones que imprimen las observaciones:

PRINT*, Nombre, Nota1,Nota2,Nota3,Promedio, 'PASS'
PRINT*, Nombre, Nota1,Nota2,Nota3,Promedio, 'FAIL'

Finalmente, puede cambiar los datos del archivo MARKS.F90 para introducir algunos nombres (no olvidar los apóstrofos):

'Alejo, Kenyo' 40 60 43
'Vladimir, Lenin' 60 45 43
'Laura, Chozo' 13 98 47

CARACTERES COMO CONSTANTE

Se ha aprendido dos tipos de datos intrínsecos: el Entero y Real. Nosotros veremos ahora el tipo de dato carácter (intrínseco).

La constante literal de carácter básico es una cadena [String] de caracteres encerrado entre un par de apóstrofos (') o doble comillas ("). La mayoría de los caracteres a utilizar son usados por su computadora, con la excepción de los "controles de caracteres" (p. ej. Escape). Los apóstrofos y las doble comillas sirven como delimitadores y no parte de la constante.

Un blanco en una constante de tipo carácter es importante, para distinguirlo, tenemos dos ejemplos:

"B Shakespeare" no es al igual que "BShakespeare"

Fortran 90 es "altamente sensible" para el caso constante carácter: ejemplo:

Carlos Tuesta no es al igual que **CARLOS TUESTA**

Hay dos maneras de representar los delimitadores de caracteres una que sea de un solo carácter como una constante de carácter. O el tipo de delimitadores puede empotrarse en una cadena delimitado por la forma corta, ejemplo:

'Jesús Dijo, " me seguiras" '

Alternativamente, el delimitado puede repetirse, como :

'Pilatos dijo, "Esto es verdad?' ' '

Una cadena de carácter puede ser vacía, es decir ' ' o " ". El número de caracteres en una cadena se conece por la declaración LENGTH. Una cadena vacía tiene una longitud de cero.

VARIABLES DE TIPO CARACTER (CHARACTER)

La Sentencia:

CHARACTER *LETRA*

Nos declara una variable llamada LETRA que almacenará un carácter de longitud 1, es decir puede retener un carácter único. Los caracteres más largos pueden declararse de la siguiente forma:

CHARACTER (Len = 15) Nombre

Esto significa que la variable tipo carácter ***Nombre*** carácter puede retener una cadena de 15 caracteres.

Una forma alternativa de la declaración es:

CHARACTER Nombre*15

Una constante de tipo carácter pueden asignarse a las variables de la manera siguiente:

Nombre = 'Jabon Jhonson '

NOMBRE DE CONSTANTES

Analizemos la siguiente declaración:

REAL, PARAMETER :: G = 9.8

Vemos que se usa para declarar G como una constante nombrada, o parámetro. El efecto de esta es que G no puede cambiarse luego de ejecutarse el programa de intentarlo se generará un mensaje de error.

El atributo PARAMETER es uno de muchos que pueden especificarse en una declaración tipo. Los atributos adicionales se introducirán luego.

Las constantes nombrados pueden ser usados como valores de inizialización. Ejemplos:

```
REAL, PARAMETER ::  Pi = 3.141593
INTEGER, PARAMETER :: Dos = 2
REAL, PARAMETER :: UnoSobre2Pi = 1 / (2 * Pi)
REAL, PARAMETER :: PiQuadrado = Pi ** Dos
```

Las expresiones de inicialización se evalúan en tiempo de compilación, hay ciertas restricciones sobre su forma.

- Se pueden involucrar con operadores intrínsecos;
- La exponenciación el operador debe tener una potencia de tipo entero.
- Las funciones intrínsecas deben tener como argumentos de tipo entero o carácter asi como sus resultados.

La siguiente es un ejemplo no permitido, dado la definición de Pi arriba:

```
REAL, PARAMETER :: UnoSobreRaiz2Pi = 1 / SQRT(2 * Pi)
```

En general, una comilla doble debe aparecer adondequiera que un atributo se especifique o una expresión de inicialización se use; de otra manera es optativo. Si el atributo PARAMETER se especifica, una expresión de inicialización debe aparecer.

Si la constante nombrado es de tipo carácter, su longitud puede declararse con un asterisco. La longitud real es determinada entonces por el compilador, ahorrando a usted la molestia de contar todos los caracteres. Por ejemplo:

```
 CHARACTER (LEN = *), PARAMETER &
 :: Mensaje = 'Presione ENTER para continuar '
```

LEN = *, constante carácter.

DATOS DE TIPO COMPLEJO : (COMPLEX TYPE)

Los números complejos y la aritmética compleja son utilizadas por Fortran 90. Por ejemplo:

```
COMPLEX, PARAMETER :: i = (0, 1)    !sqrt(- 1)
COMPLEX X, Y
X = (1, 1)
Y = (1, - 1)
PRINT*, CONJG(X), I * X * Y
```

La salida es:

```
(  1.0000000, - 1.0000000) (  0.0000000E+00,  2.0000000)
```

Cuando una constante complejo es ingresada con READ* debe adjuntarse entre parentesis. Muchas de las funciones intrínsecas pueden tomar argumentos complejos.

INTRODUCCION A LAS FUNCIONES INTRINSECAS

Fortran 90 contiene una diversidad de funciones intrinsecas capaz de evaluar en forma rápida a traves de una serie de argumentos en diversas operaciones tales como aritméticas simples.

Así mismo las funciones matemáticas especiales, cosenos, logaritmos, etc.

MOVIMIENTO DE UN PROYECTIL :

Nosotros queremos escribir un programa para computar la posición (X,Y) coordenadas y la velocidad (magnitud y la dirección) de un proyectil, dado un tiempo t, el tiempo de

lanzamiento, u, la velocidad de lanzamiento, desde un ángulo inicial de lanzamiento (en grados), siendo g, la aceleración debido a la gravedad.

Los desplazamientos horizontales y verticales son dados por las fórmulas:

$x = u.t.Cos(a)$ $y = u.t.\sin(a) - g.t^2/2$

La magnitud de la velocidad V es: $V = \sqrt{V_x^2 + V_y^2}$, donde sus componentes horizontales y verticales, son dados por V_x , V_y .

$V_x = U.Cos.a$ $V_y = U.Sin.a - g.t$ y V hace un ángulo θ con el terreno tal que .

$$Tan.\theta = \frac{V_Y}{V_x}$$

El programa es:

```
PROGRAM Proyectil
IMPLICIT NONE
REAL, PARAMETER :: g = 9.8            !aceleración debido a la gravedad
REAL, PARAMETER :: pi = 3.1415927   !constante conocida
REAL A          !lanzamiento,  ángulo en grados
REAL T          !tiempo de vuelo
REAL Theta    !la dirección en grados en un tiempo T
REAL  U         !velocidad de lanzamiento
REAL V          !velocidad resultante
REAL Vx         !velocidad horizontal
REAL Vy         !velocidad vertical
REAL X          !desplazamiento horizontal
REAL Y          !desplazamiento vertical
READ*, A, T, U
A = A * Pi / 180       ! convierta ángulo a radianes
X = U * COS( A ) * T
Y = U * SIN( A ) * T - g * T * T / 2.
Vx = U * COS(A )
Vy = U * SIN( A ) - g * T
V = SQRT( Vx * Vx + Vy * Vy )
Theta = ATAN( Vy / Vx ) * 180 / Pi
PRINT*, 'x: ', X, 'y: ', Y
PRINT*, 'V: ', V, 'Theta: ', Theta
END
```

Si usted ejecuta este programa con los siguientes datos :

45 6 60

La salida en pantalla:

```
45 6 60
x:     254.558400          y:        78.158440
V:      45.476310          Theta:   -21.103060
Press any key to continue
```

Usted observa el valor negativo de **Theta** nos indica que el proyectil viene bajando. El argumento de una función puede ser otra función teniendo una expresión de tipo apropiado.

La fórmula de V podría haber sido computada directamente como se indica a continuación:

```
V = SQRT( (U * COS( A )) ** 2 + ( U * SIN( A ) - g * T) ** 2 )
```

Los ángulos para las funciones trigonometricas deben expresarse en radianes, y devuelven radianes. Para convertir grados a radianes, multiplicando el ángulo en grados por pi/180, donde pi es el número trascendental bien conocido 3.1415926....

FUNCIONES INTRINSECAS EN FORTRAN 90

1. ABS(X) :Valor absoluto sea X entero,real o complex.

2. ACOS(X) :Arco Coseno de X.

3. ASIN(X) :Arco Seno de X

4. ATAN(X) :Arco Tangente de X en el rango - pi/2 a pi/2.

5. ATAN2(Y, X) :Arco Tangente Doble de y/x en el rango - pi a +pi.

6. COS(X) :Coseno de X real o complex.

7. COSH(X) :Coseno hiperbólico de X.

8. COT(X) :Cotangente de X.

9. EXP(X) :Valor de la exponencial e_x, donde X real o complejo.

10. INT(X [,**TIPO**]) :Convierte a entero, donde X puede ser real o complejo truncando la parte decimal hacia cero, por ejemplo: INT(3.9) devuelve 3, INT(- 3.9) retorna –3. Si **TIPO** del argumento está presente, especificar el valor del parámetro . De otra manera el resultado tiene tipo de entero.

11.LOG(X) :Logaritmo natural de X siendo real o complejo. Si el argumento es entero ocasionará un error.

12.LOG10(X) :Logaritmo de base 10.

13.MAX(X1, X2[, X3, ...]) :Devuelve el valor máximo de dos o más entero o reales.

14.MIN(X1, X2[, X3, ...]) :Devuelve el valor mínimo de dos o más entero o reales.

15.MOD(K, L) :Devuelve el residuo cuando K es dividida por L. Los argumentos de ambos enteros o ambos reales.

16.NINT(X [,TIPO]) :Devuelve el entero más cercano a X, por ejemplo: NINT(3.9) devuelve 4, mientras NINT(- 3.9) devuelve - 4.

17.REAL(X [,TIPO]) :La función real convierte a entero, X real o complejo al tipo real, por ejemplo: REAL(2)/4 devuelve 0.5, considerando REAL(2/4) devuelve 0.0.

18. SIN(X) : Seno de X , X real o complejo.

19.SINH(X) :Seno hiperbólico de X.

20.SQRT(X) :Raíz cuadrada de X , real o complejo.

21.TAN(X) :La tangente de X.

22.TANH(X) :Tangente hiperbólico de X.

Subrutinas Intrínsecas: (Procedimientos estandars)

Fortran 90 contiene un número de subrutinas intrínsecas. Las subrutinas difieren ligeramente de las funciones dado que las subroutinas se invocan con una declaración **CALL**, y los resultados se devuelven mediante argumentos de entrada. En cambio las funciones pueden estar referenciadas en operaciones/expresiones aritmeticas.

El ejemplo muestra como usted puede mostrar la fecha y la hora.

```
PROGRAM FECHA_HORA
CHARACTER FECHA*8,HORA*8
CALL DATE_AND_TIME(FECHA,HORA) !Subroutina DATE_AND_TIME
PRINT*,FECHA
PRINT*, HORA(1:2)//':'//HORA(3:4)//':'//HORA(5:10)
END PROGRAM FECHA_HORA
```

La salida del programa en la pantalla es:

```
20001125
07:46:52.9
Press any key to continue
```

La Declaración GO TO

Esta sentencia permite desviar el control del programa a otro punto. GO TO es un declaracion incondicional, y la sintáxis:

GO TO *etiqueta*

Donde:

***etiqueta**:* es un número de declaración que aparece al incio de una proposión del programa: el número estara en el rango: 1 – 99999 precediendo una declaración sobre la misma línea.

Ejemplo:.

```
    GO TO 99
    X = 67.8
99  Y = - 1
```

La sentencia X = 67.8 nunca sera ejecutado, a menos que se ocasione algo para abortar el programa.

Considere el siguiente segmento de código (L1 y L2 son dos variables lógicos definidos):

```
IF (L1) THEN
  I =1
 J = 2
ELSE IF  (L2) THEN
  I  = 2
 J = 3
ELSE
 I = 3
```

```
 J = 4
END IF
```

Si prescindimos del IF lógico, usando GO TO se debe construir la codificacion siguiente:

```
      IF (.NOT.L1) GOTO 10
      I = 1
      J  = 2
      GOTO 30
 10   IF (.NOT.L2) GOTO 20
      I  = 2
      J = 3
      GOTO 30
 20   I  = 3
      J = 4
 30   CONTINUE
```

EJERCICIOS RESUELTOS Y PROPUESTOS:

1. Escriba un programa que lea dos números (pueden ser iguales) y evalue quien el mayor de estos dos, indicando con un mensaje adecuado, y si son iguales ambos, que escriba un mensaje para tal efecto.

Solución:

```
Program Mayor_de_dos_Numeros
Real  A, B
Read*, A,B

IF ( A>B ) Then
        Print*, A, "Es Mayor"
Else If (B>A) Then
        Print*, B, "Es Mayor"
Else
        Print*, A, B, "Son iguales"
End If
End Program Mayor_de_dos_Numeros
```

2. Escribir un algoritmo para elaborar un programar para el problema siguiente: "Leer 10 enteros y escriba cuántos de ellos son positivos, negativos o cero. Escriba el algoritmo con la estructura de decisión múltiple IF, y luego diseñe el programa usando la declaración CASE.

Solución: Primera parte (algoritmo → Pseudocodigo)

```
Repetir 10 Veces
        Si numero < 0
        Entonces
        Incremente al contador de negativos
        SiNO Si numero = 0
        Entonces
        Incremente al contador cero
        SINO
        Incremente al contador positivo
```

Solución: Segunda Parte:

```
Integer i,Num,Npos,Nzero,Nneg
Npos=0;Nzero=0;Nneg=0;
Do I =1,10
  Read*,Num
        Select Case (Num)
        Case(: -1)
        Nneg = Nneg+1
        Case(0)
        Nzero = Nzero+1
        Case Default
        Npos = Npos+1
        End Select
End Do
Print*,Nneg,Nzero,Npos
End
```

```
 0
-1
 0
 2
-3
 0
-1
 0
 5
-1
        4         4         2
Press any key to continue
```

3. Desarrolle un pseudocodigo que de solución a dos ecuaciones lineales simultáneas. El algoritmo debe ser capaz de manejar todas las situaciones posibles, es decir rectas que se cruzan, paralelas. Además escriba un programa para implementar su algoritmo, y pruebe sobre algunas ecuaciones para que usted observe las soluciones:

x + y = 3
2x–y = 3
(x = 2, el y = 1). Coincidentes: comience por derivar una fórmula algebraica para la solución del sistema
ax + by = c
dx + ey = f
El programa deberá leer los coeficientes ***a, b, c, d, e, f.***
Solución: Primera Parte:

1. Lea a,b,c,d,e,f
2. u =ae – db
3. v = ec – bf
4. Si u = 0 y v =0 Entonces
   Escribir “Rectas son coincidentes”
   SiNo Si u =0 y v no es igual a 0 Entonces
   Escribir “Rectas son paralelas”
   SiNo
   X = v/u
   Y = (af-dc)/u
   Imprima “X,Y”, X,Y
   Fin de Si

Solución: Segunda Parte:

```
Program rectas

Real A,B,C,D,E,F,U,V,X,Y
Read*, A,B,C,D,E,F
U = A*E-D*B
V = E*C-B*F
IF (U == 0 .AND. V==0) THEN
  PRINT*,"RECTAS SON COINCIDENTES"
  ELSE  IF (U == 0 .AND. V /= 0) THEN
   PRINT*, "RECTAS PARALELES"
          ELSE
```

```
1 1 3 2 -1 3
X,Y       2.000000        1.000000
Press any key to continue

5 5 5 5 5 5
RECTAS SON COINCIDENTES
Press any key to continue
```

```
        X=V/U
        Y=(A*F-D*C)/U
        PRINT*,"X,Y",X,Y
        END IF
End Program rectas
```

4. Supongamos que la prima por seguro de salud se descuenta del salario de un empleado, de acuerdo con el siguiente plan:
Prima es igual 9.75 si es soltero
Prima = 16.25 si es casado y sin hijos
Prima = 24.50 si es casado y con hijos
Elabore un programa que calcule el sueldo neto real, según las condiciones establecidas.

5. La comision sobre las VENTAS totales de un empleado es como sigue:
Si VENTAS es menor a $ 50, entonces no hay comisión
Si Ventas es mayor igual a $ 50 pero menor igual que $ 550, entonces comision es igual a 10% de las Ventas.
Si las Ventas es mayor $ 500, entonces la comision es igual a $ 50 + 8% de las ventas superiores a $ **500.**

CAPITULO IV

EJEMPLOS BASICOS DE LA DECLARACION – DO

Ejemplo 1:

```
INTEGER I
DO I = 1,10,1
  PRINT*,2*I
END DO
PRINT*,I
END
```

```
 2
 4
 6
 8
 10
 12
 14
 16
 18
 20
 11
Press any key to continue
```

Ejemplo 2:

```
INTEGER I,J
PRINT*,"INGRESE VALOR DE I,J"
READ*,I,J
    DO L = I,J,1
   PRINT*,2*L
   END DO
END
```

```
INGRESE VALOR DE I,J
1  6
 2
 4
 6
 8
 10
 12
Press any key to continue
```

Ejemplo 3:

```
INTEGER I,J,K
PRINT*,"INGRESE VALOR DE I,J,K"
READ*,I,J,K
    DO L = I,J,K
     PRINT*,L
    END DO
END
```

```
INGRESE VALOR DE I,J,K
3  10  3
 3
 6
 9
Press any key to continue
```

Ejemplo 4:
Programa que determina la serie de Fibonacci: 1,1,2,3,5,8,13,21,34,…..

```
INTEGER :: FIB1,FIB2,N,NUEVO
PRINT*,"INGRESE VALOR N"
READ*,N
IF (N<3) THEN
   PRINT*, "ERROR"
ELSE
  FIB1=1
  FIB2=1
  PRINT*,FIB1,FIB2
    DO I = 3,N
    NUEVO=FIB1+FIB2
   FIB1=FIB2
   FIB2=NUEVO
   PRINT*,NUEVO
   END DO
END IF
END
```

```
INGRESE VALOR N
10
 1          1
 2
 3
 5
 8
 13
 21
 34
 55
Press any key to continue
```

Ejemplo 5:

```
INTEGER I,J,EXTERNO,INTERNO
PRINT*,"INGRESE VALOR I,J"
READ*,I,J
DO EXTERNO = 1,I
 DO INTERNO= 1,J
 PRINT*,EXTERNO,INTERNO,EXTERNO*INTERNO
 END DO
END DO
END
```

```
INGRESE VALOR I,J
 2  4
 1   1   1
 1   2   2
 1   3   3
 1   4   4
 2   1   2
 2   2   4
 2   3   6
 2   4   8
Press any key to continue
```

El DO CONDICIONAL (The conditional Loop)

La estructura ***DO WHILE*** es un tipo de bucle condicional. El bucle se ejecutará indefinidamente hasta que la prueba se condicione a FALSO, con base a la variable de control del bucle(s).

Por supuesto, para comenzar el bucle, la condición debe inicialmente ser verdadera. Obviamente, la condición debe cambiar de algún modo de verdadero a falso. Esta es una diferencia con la declaración **DO**, donde la computadora no nos permite cambiar el control de la variable.

Para el ***DO WHILE***, la variable de control(s) que es la base de la condición de prueba debe cambiar, de otra manera, nosotros se estara inmerso en un bucle infinito
El formato de la declaración DO WHILE es:

DO WHILE <Condición es Verdadera>
[*bloque de instrucciones*]
END DO

Cuando se ingresa al bucle primero, la prueba se realiza para lo cual los resultados permitidos únicos son verdaderos o falsos. Si la condición es verdadera, el bloque de instrucciones es ejecutado. Pero si la condición de prueba es falsa, el bucle termina y realiza un salto la operación después del fin del bucle

Ejemplo1

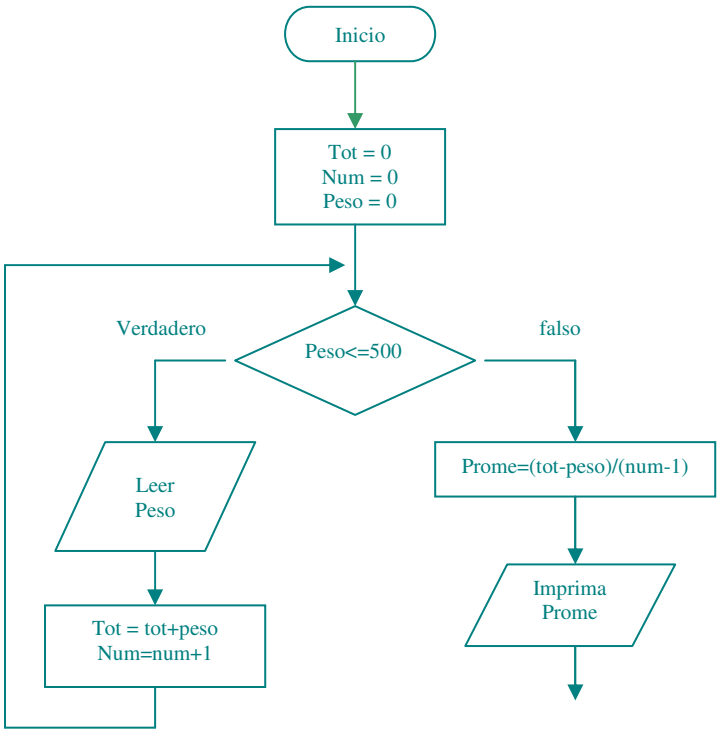
Una demostración de como trabaja la declaración WHILE DO, se presume que nosotros deseamos calcular el promedio de peso de los conejos en un laboratorio. Donde los conejos se multiplican tan rapidamente, nosotros nunca sabemos por adelantado cuántos habrá. Nosotros establecemos el bucle para leer los pesos, de uno en uno, hasta que uno de los pesos sea mayor de 500 libras. Cuando esto ocurre, el bucle parará. Nosotros a veces llamamos este valor especial como Valor Centinela. El bucle se establece para que nosotros miremos a este valor como clave,

```
!Nosotros usaremos 500 como centinela
!Valor de control del loop
TOT = 0.0
NUM = 0
PESO =0.0
DO WHILE (PESO<=500.0)
   PRINT*,"Ingrese peso"
   READ*, PESO
  TOT = TOT+PESO
  NUM = NUM+1
END DO
PROME=(TOT-PESO)/(NUM-1)
```

```
Ingrese peso
100
Ingrese peso
200
Ingrese peso
300
Ingrese peso
400
Ingrese peso
500
Ingrese peso
600
Promedio de Pesos=     300.000000
Press any key to continue
```

```
PRINT*, "Promedio de Pesost=",PROME
END
```

Diagrama de Flujo:



Las Declaraciones EXIT y CYCLE

Fortran 90 también ofrece un propósito general que puede usarse como un contador o un bucle condicional, la forma general de la sináxis siguiente : (lo que esta entre corchetes es opcional).

```
                 DO [Etiqueta]  [Control de estructura de enlaze]
                 [ IF (Condición es verdadera )  EXIT]
                 [ IF (Condicion es verdadera )  CYCLE]
[Etiqueta]       END DO
```

Si la variable de control del bucle se incluye sobre la misma línea como la sentencia **DO**, el bucle llega a ser un bucle contador. Pero si los variables de control del bucle son fuera de la izquierda, el bucle llega a ser de tipo condicional.

En este caso, sin embargo, debe haber un medio de parar el bucle. Estos se hecho con dos nuevos comandos: EXIT y CYCLE.

Como sus nombres implican, la declaración EXIT dentro del bucle ocasiona que dicho bucle termine, mientras el comando CYCLE ocasiona que el bucle salte al próximo valor de **LCV**. En efecto, ambas declaraciones son los que sustituyen a la sentencia GO TO. El comando EXIT es equivalente al declaración GO TO a un punto fuera del bucle. Similarmente, el comando CYCLE es equivalente al GO TO al final del bucle.

Ejemplo 2: Modificación del ejemplo1usando la sentencia EXIT

```
!Nosotros usaremos 500 como centinela
!Valor de control del loop
TOT = 0.0
NUM = 0
PESO =0.0
DO
  PRINT*,"Ingrese peso menor a 500"
  READ*, PESO
```

```
Ingrese peso menor a 500
100
Ingrese peso menor a 500
200
Ingrese peso menor a 500
300
Ingrese peso menor a 500
500
Ingrese peso menor a 500
600
Promedio de Pesos=     275.000000
Press any key to continue
```

```
  IF (PESO > 500.0) EXIT
  TOT = TOT+PESO
  NUM = NUM+1
END DO
PROME=TOT/NUM
PRINT*, "Promedio de Pesos=",PROME
END
```

Ejemplo3: Modificación del ejemplo1 usando la sentencia CYCLE.

```
!Nosotros usaremos 500 como centinela
!Valor de control del loop
TOT = 0.0
NUM = 0
PESO =0.0
DO
PRINT*,"Ingrese peso"
READ*, PESO
IF (PESO > 500.0) EXIT
IF (PESO<=0.0) THEN
PRINT*,"Error en ingreso de dato, ingrese nuevo valor"
 CYCLE
 ELSE
 TOT = TOT+PESO
 NUM = NUM+1
 END IF
 END DO
PROME=TOT/NUM
PRINT*, "Promedio de Pesos=",PROME
END
```

```
Ingrese peso
200
Ingrese peso
100
Ingrese peso
300
Ingrese peso
-100
Error en ingreso de dato, ingrese nuevo valor
Ingrese peso
400
Ingrese peso
600
Promedio de Pesos=     250.000000
Press any key to continue
```

Ejemplo4:

```
    INTEGER :: FT
    DO FT = 0, 4
    DO IN = 0,3
    INTOT = IN + FT*12
    PRINT 5,FT,IN, INTOT
    END DO
    END DO
5   FORMAT('', I3, '  Feet and', I3, ' Inches =', I5 ,' Inches')
    END
```

```
 0 Feet and  0 Inches =    0 Inches
 0 Feet and  1 Inches =    1 Inches
 0 Feet and  2 Inches =    2 Inches
 0 Feet and  3 Inches =    3 Inches
 1 Feet and  0 Inches =   12 Inches
 1 Feet and  1 Inches =   13 Inches
 1 Feet and  2 Inches =   14 Inches
 1 Feet and  3 Inches =   15 Inches
 2 Feet and  0 Inches =   24 Inches
 2 Feet and  1 Inches =   25 Inches
 2 Feet and  2 Inches =   26 Inches
 2 Feet and  3 Inches =   27 Inches
 3 Feet and  0 Inches =   36 Inches
 3 Feet and  1 Inches =   37 Inches
 3 Feet and  2 Inches =   38 Inches
 3 Feet and  3 Inches =   39 Inches
 4 Feet and  0 Inches =   48 Inches
 4 Feet and  1 Inches =   49 Inches
 4 Feet and  2 Inches =   50 Inches
 4 Feet and  3 Inches =   51 Inches
Press any key to continue
```

Ejemplo5:

```
REAL :: NUM
INTEGER :: TERM
NUM = 2.0
DEN = 1.0
DO TERM = 1,10
VAL = NUM/DEN
PRINT*,TERM,'#',NUM,'/',DEN,'= ',VAL
PRENUM = NUM
```

```
 1#      2.000000  /      1.000000 =       2.000000
 2#      3.000000  /      2.000000 =       1.500000
 3#      5.000000  /      3.000000 =       1.666667
 4#      8.000000  /      5.000000 =       1.600000
 5#     13.000000  /      8.000000 =       1.625000
 6#     21.000000  /     13.000000 =       1.615385
 7#     34.000000  /     21.000000 =       1.619048
 8#     55.000000  /     34.000000 =       1.617647
 9#     89.000000  /     55.000000 =       1.618182
 10#   144.000000  /     89.000000 =       1.617977
Press any key to continue
```

```
PREDEN = DEN
NUM = PRENUM + PREDEN
DEN = PRENUM
END DO
END
```

DECLARACION → CONTINUE

Este comando no afecta el desarrollo del programa. Se usa como final de bloque para sentencias tipo DO WHILE.

Sintáxis:

CONTINUE

Ejemplo:

```
      DO 10 J = 1,10
      WRITE(*,*) J
10          CONTINUE
      DO 20 I = 1,5
      WRITE(*,*) I*I
20          CONTINUE
      END
```

```
1
2
3
4
5
6
7
8
9
10
1
4
9
16
25
Press any key to continue
```

SENTENCIA → FORMAT :

Esta sentencia fija el formato en el cual los datos serán leidos o escritos. Está necesariamente asociado a un comando de entrada/salida de datos.

Sintáxis:

<Etiqueta> Format (ListaEdición)

Donde:

ListaEdición: Permite especificar características como espacios en blanco, lineas en blanco, números de posiciones enteras, etc.

Iw Para números Enteros **w** indica el número de posiciones, incluyendo el signo. I5
Fw.d Para números Reales. **W** especifica el tamaño total incluyendo signo, punto y decimales, **d** indica el número de posiciones decimales. F6.2
Ew.d Para números Reales en formato exponencial. E12.4
Dw.d Doble presición D20.8
Aw Para caracteres, **w** indica el número de posiciones., A20
‘’ Cadena de caracteres, ‘xxx’, ‘Ejemplo’
nX Representa espacios en blanco, **n** especifica el número de espacios en blanco.
/ Salto de linea

COMPOSICION DE LAS SENTENCIAS DE ENTRADA/SALIDA CON LA SENTENCIA FORMAT

a) Entrada de Datos:

READ **sl**, var1,var2,…..
sl FORMAT (Lista de Instrucciones)

b) Salida de Datos:

PRINT **sl**, var1,var2,…..
sl FORMAT (Lista de instrucciones)

sl FORMAT (CCC,Especif1,Especif2,…..EspecifN)
Donde:
sl = Sentencia de etiqueta
CCC = Control de caracteres
Specif = Instrucción para cada variable en forma individual.

Carácter	**Descripción de Funciones**
‘ ’	Espacio vertical simple
‘0‘	Espacio vertical doble
‘1’	Nueva página
‘+ ‘	No avance, regresa al inicio de la actual linea.

Ejemplos Básicos:

```
    ICOUNT = 237
    JCOUNT = -14
    PRINT 33, ICOUNT,JCOUNT
33  FORMAT (',I6,I9)
    END
```

```
 237    -14
Press any key to continue
```

```
    ICOUNT = 12345
    JCOUNT = -98765
    PRINT 98, ICOUNT, JCOUNT
98  FORMAT (',I5,I5)
    END
```

```
2345*****
Press any key to continue
```

Nota: En la salida de la pantalla vemos que el ancho es demasiado pequeño para la salida de las variables ICOUNT,JCOUNT, modifique el programa.

```
    DIST = 12.345
    TIME = 0.00345
    VELOC = DIST/TIME
    PRINT 5,DIST,TIME,VELOC
5   FORMAT (',F7.2,F9.6,F10.1)
    END
```

```
12.35  .003450    3578.3
Press any key to continue
```

```
    DIST = 12.345
    TIME = 0.00345
    VELOC = DIST/TIME
    PRINT 5,DIST,TIME,VELOC
5   FORMAT (',E12.4,E14.6,F10.1)
    END
```

```
 .1235E+02   .345000E-02    3578.3
Press any key to continue
```

```
    DOUBLE PRECISION :: DIST,TIME,VELOC
    DIST = 12.345
    TIME = 0.00345
    VELOC = DIST/TIME
    PRINT 5,DIST,TIME,VELOC
5   FORMAT (',D12.4,D11.3,D10.1)
    END
```

```
 .1235D+02   .345D-02    .4D+04
Press any key to continue
```

```
Martin Cwiakala
Press any key to continue
```

```
      CHARACTER :: NAME*20
      NAME = 'Martin Cwiakala'
      PRINT 19,NAME
19    FORMAT ('',A21)
      END
```

Ing. William Chauca Nolasco
96933582

```
      PRINT 21
21    FORMAT (' ', 'INGRESE X, Y, Z:')
      X=12.34
      Y=-0.025
      PRINT 34,X,Y,X*Y
34    FORMAT(' ','X = ',F7.2,' Y = ',F7.3, '  PROD = ', F10.5)
      END
```

```
INGRESE X, Y, Z:
X =  12.34 Y =  -.025  PROD =   -.30850
Press any key to continue
```

```
      X=1.234567
      Y=9.876543
      PRINT 10,X,Y
10    FORMAT(' ','Valor de X=',F8.3,3X,'Valor de Y=',F5.1)
      END
```

```
Valor de X=  1.235   Valor de Y=  9.9
Press any key to continue
```

```
      BASE =12.4
      HEIGHT=9.6
     VOL=119.04
     PRINT 9, BASE, HEIGHT,VOL
9    FORMAT (' ',5X,F9.3,/,3X,' X',T7,F9.3,/,T6,'_________',/,T7,F9.3)
     END
```

```
    12.400
X   9.600
_________
  119.040
Press any key to continue
```

Ejemplo:

```
      TEMP1 = 5.0
      TEMP2 = 10.0
      TEMP3 = 15.0
      TEMP4 = 20.0
      TEMP5 = 25.0
      X1 = 12.400
      X2 = 13.736
      X3 = 13.055
      X4 = 13.343
      X5 = 13.587
      EXPAN1 = 0.00
      EXPAN2 = 2.710
      EXPAN3 = 2.505
      EXPAN4 = 2.206
      EXPAN5 = 1.829
      PRINT 23
      PRINT 24
      PRINT 19,TEMP1,X1,EXPAN1
      PRINT 19,TEMP2,X2,EXPAN2
      PRINT 19,TEMP3,X3,EXPAN3
      PRINT 19,TEMP4,X4,EXPAN4
      PRINT 19,TEMP5,X5,EXPAN5
23    FORMAT(",T5,"TEMP(C)",T17,"LENGTH-BAR(Cm)",T36,"EXPANSION(%)")
24    FORMAT (",TL35,13('_'),TR2,15('_'),TR5,13('_'))
19    FORMAT(",T5,F7.1,T15,F11.3,T35,F9.3)
      END
```

TEMP (C)	LENGTH-BAR(Cm)	EXPANSION(%)
5.0	12.400	.000
10.0	13.736	2.710
15.0	13.055	2.505
20.0	13.343	2.206
25.0	13.587	1.829

Press any key to continue

FUNCIONES DE CADENA DE CARACTERES

- LEN(CADENA): da el número de caracteres en LA CADENA si es escalar, o un elemento de cadena si es un array.

- REPEAT(CADENA, NCOPIAS): Concatenan NCOPIES de una CADENA; ambos argumentos deben ser escalar.

- TRIM(CADENA): CADENA (escalar) quita o remueve blancos en una cadena de caracteres.

Note que los argumentos pueden ser reales o complejos, escalares o arrays, a menos que dé otra cosa.

- AIMAG(Z): Devuelve la parte imaginaria.
- AINT(A [,KIND]): Devuelve el número real no más grande de su argumento, ejemplo: AINT(3.9) retorna 3.0.
- ANINT(A [,KIND]): Devuelve el número real más cercano, por ejemplo: ANINT(3.0) retorna 4.0.
- CEILING(A): Devuelve el entero más pequeño no menos de A.
- CMPLX(X [,Y] [,TIPO]): convierte X o (X, Y) al tipo complejo.
- CONJG(Z): conjugada de Z complejo.
- DIM(X, Y): max(X-Y, 0).
- FLOOR(A): Retorna el entero más grande no excediendo a su argumento, ejemplo:. FLOOR(-3.9) retorna -4.

Ejemplo:

Este primer ejemplo utiliza un Array y los edita en un descriptor para imprimir caracteres.

Suponga que nosotros queremos analizar los resultados de una prueba escrita por una clase de estudiantes. Quisiéramos saber cuántos estudiantes obtuvo el porcentaje dentro del rango: 0-9, 10-19, ..., 90-99. Cada uno de estos rangos se les llama, numerado desde cero para la conveniencia. Nosotros también necesitamos proveer para los rangos mas altos que consigue 100. Suponer los números de estudiantes quien consiguen marcas en estos rangos son como se indica a continuación:

1 0 12 9 31 26 49 26 24 6 1

Es decir 12 marcas obtenidas en el rango 20-29. Nosotros necesitamos un conjunto F(0:10), decir, 11 elementos, donde cada elemento almacena el número de estudiantes con marcas en un rango particular, por ejemplo. F(2) debería tener el valor 12. El programa siguiente imprime un mapa de barra de la frecuencia distribución F, donde cada asterisco representa un estudiante en que rango se encuentra:

```
      INTEGER, DIMENSION(0:10):: F=(/ 1,0,12,9,31,26,49, 26,24, 6, 1 /)
10    FORMAT( I3, ' - ', I3, '  (', I3, '):', 60A1 )
20    FORMAT( '100', 6X, '  (', I3, '):', 60A1 )
      DO I = 0, 10
      IF (I < 10) THEN
      PRINT 10, 10 * I, 10 * I + 9, F(I), ('*', J = 1, F(I))
      ELSE
      PRINT 20, F(I), ('*', J = 1, F(I))
      END IF
      END DO
```

END

```
 0 -  9 (  1):*
10 - 19 (  0):
20 - 29 ( 12):**********
30 - 39 (  9):*********
40 - 49 ( 31):*******************************
50 - 59 ( 26):**************************
60 - 69 ( 49):*************************************************
70 - 79 ( 26):**************************
80 - 89 ( 24):************************
90 - 99 (  6):******
00      (  1):*
Press any key to continue
```

Observe la ausencia de asteriscos para el rango 10-19.
Esto es porque F(1) ha tenido el valor cero, que implica la declaracion DO tiene la declaración PRINT recepciona el cero cuando "I" toma el valor 1.

Por supuesto, en una situación verdadera, las frecuencias no estarán presente prolijamente sobre un plato. Es más probable tener una lista de las marcas reales.

Usted debería adaptar el programa para leer un conjunto de muestreo de marcas, en el rango 0-100, y luego convertirlos en frecuencias. El mecanismo básico es:

```
READ( ... ) MARK
K = INT( MARK / 10 )      ! K is the decile
F(K) = F(K) + 1           ! another mark in the Kth decile
```

Ejemplo:

```
IMPLICIT NONE
CHARACTER (10) Word1, Word2
READ*, Word1
READ*, Word2
IF (Word1 < Word2) THEN
PRINT*, Word1, Word2
ELSE
PRINT*, Word2, Word1
END IF
CALL ToUpper( Word1 )
CALL ToUpper( Word2 )
IF (Word1 < Word2) THEN
PRINT*, Word1, Word2
ELSE
PRINT*, Word2, Word1
END IF
CONTAINS
SUBROUTINE ToUpper( String )
CHARACTER (LEN = *) String
INTEGER I, Ismall, IBIG
Ismall = ICHAR( 'a' )
IBIG = ICHAR( 'A' )
DO I = 1, LEN( String )
IF (String(I:I) >= 'a' .AND. String(I:I) <= 'z') THEN
```

```
maria
luis
luis        maria
LUIS        MARIA
Press any key to continue
```

```
String(I:I ) = CHAR( ICHAR( String(I:I) ) + IBIG – Ismall )
END IF
END DO
END SUBROUTINE
END
```

CAPITULO V

ESTRUCTURAS DE DATOS :

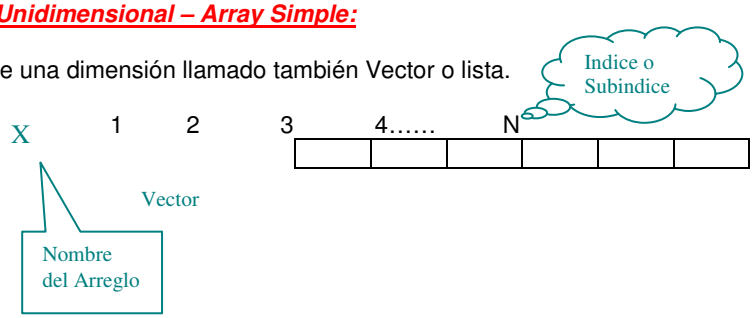
Estructuras Simple: Arrays ó Arreglos:

Arreglo: Es un conjunto finito y ordenado de elementos homogéneos (mismo tipo de dato).

- Finito: Siempre será necesario especificar el número de elementos que tiene el arreglo.
- Ordenado: Qué sea posible identificar el primero, segundo,……..n-ésimo elemento del arreglo.
- Homogéneo: todos los elementos son del mismo tipo.
- Se almacenan normalmente en posiciones contiguas de la memoria a partir de una dirección inicial.

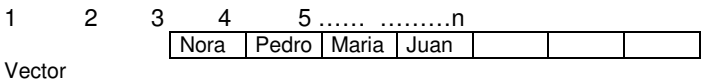
Arreglo Unidimensional – Array Simple:

Arreglo de una dimensión llamado también Vector o lista.



Indice o subíndice: Indica la posición de cada elemento en el arreglo

Los valores en un arreglo se guardan en cada una de las celdas.



Para referenciar a una celda, se usa el Nombre del arreglo y el índice que indica su posición relativa dentro del arreglo.
Ejemplo:
Vector[4] hace referencia al contenido de la celda “Juan”

Características principales de un arreglo:

- El menor valor de un índice de un arreglo se denomina límite inferior.
- El mayor valor de un índice de un arreglo se denomina límite superior.
- La cantidad de elementos en el arreglo se denomina rango.
- En los arreglos, al ser de tamaño finito y predefinido, no es posible incluir ni eliminar elementos, sólo se puede alterar los contenidos de un elemento.
- Las primitivas básicas para un arreglo son:
  - Creación
  - Almacenamiento
  - Extracción (acceso)
- Operación Acceso. Se realiza a través del nombre del arreglo y entre paréntesois el valor del índice.
  Ejemplo: X(I)

Primitivas:

- Creación de un arreglo; normalmente todos los lenguajes poseen una instrucción que permite crear un arreglo, en donde se indica:
  - Nombre del arreglo
  - Número de índices
  - Rango para cada índice
- Se asume que al momento de la creación los valores contenidos en el arreglo no estan definidos. Ejemplo Arreglo Vector *Dimension[1..20]*
- La operación de almacenamiento acepta un arreglo **A** un índice **i** y un valor **x** . el resultado es que el valor **x** queda almacenado en la posición **i** del arreglo **A**

- Se realiza usando el operador de asignación y entregando un valor que será guardada en una posición del arreglo.
  Ejemplo: x[10] ← 524, *almacena el valor 524 en la posición 10 del arreglo.*
- Antes de asignar un valor a un elemento del arreglo, su valor no está definido.
- La extracción es una función que acepta el nombre del arreglo **A** y un índice **i** y retorna el valor contenido en la posición **i** del arreglo **A**
- Se realiza a través de una asignación de valores:

  Ejemplo: ***Nueva*** = X[4], donde el contenido de la posición 4 del arreglo se extrae y se almacena en la variable ***Nueva***
- Cuando se usan arreglos se deben tomar los siguientes cuidados:
  - Dar valor inicial a los elementos del arreglo.
  - Especificar siempre el valor del índice
  - Cuidar que los índices no tomen valores fuera de su rango.

Ejemplos:

- Almacenar los nombres de los alumnos
- Almacenar para cada día el total de ventas
- Almacenar la temperatura más alta/baja del día
- Almacenar el menú semanal (plato principal)
- Almacenar el gasto diario
- Para cada día del mes almacenar el día (lunes,martes,etc)
- Para un día realizar levantamiento de temperatura cada hora y almacenar el valor. Determine la temperatura más alta y más baja del día.
- Almacenar cada alumno y su nota. Determine el nombre del alumno con la nota más alta y la mas baja. Igualmente determine el promedio de notas.

Arreglos(Arrays) de dos dimensiones:(Matriz/Tablas)

Arreglos de dos dimensiones:

Sea la matriz **B**

B

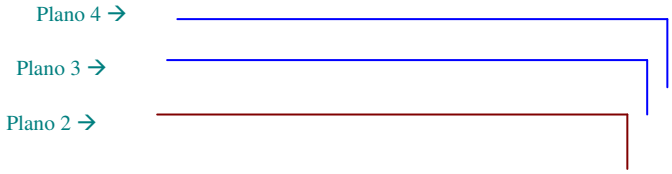
	1	2	3	4 …….		N
1						
2						
3						
.				B(I,J)		
.						
.						
M						

- Se puede considerar que un arreglo de dos dimensiones es un arreglo de una dimensión, en donde cada elemento es a su vez otro arreglo.
- Cada posición en arreglo bidimensional pertenece a una fila y a una columna.
- Para referenciar una posición es un arreglo de dos dimensiones es necesario indicar además del Nombre del arreglo un índice para la fila y otro para la columna.
- Ejemplo: **B[2,5]**

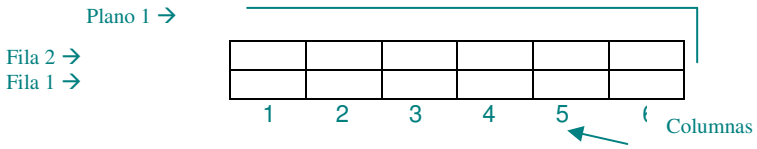
PRIMITIVAS:

- Creación: se deberá declarar el nombre del arreglo, y los rangos para las dos dimensiones.
- Ejemplo: Arreglo B, ***Dimensión [1..10,1..20]***
- Almacenamiento se realiza asignando un valor a una determinada posición del arreglo: **B [5,8]** ← "Manuela"
- Es necesario tener la seguridad de que los valores usados para los índices sean válidos, es decir que estén entre los valores límites definidos para el arreglo.
- Extracción: Se realiza a través del nombre del arreglo y entre paréntesis el valor de los índices, considerando que el primero indica el número de fila y el segundo el número de columna del elemento. Ejemplo: VIEJA ← B[i,j]

Arreglo Multidimensional:



Ing. William Chauca Nolasco
96933582



Ejemplos:

- Almacenar la temperatura cada hora para 30 días
- Determine la temperatura mínima y máxima del mes
- Almacene las notas de los 33 alumnos (4 notas por alumno). Calcular el promedio. Determine para cada calificación la nota más alta y la más baja.

Para usar un arreglo hay que declararlo en una sentencia **DIMENSION**, es necesario especificar:

- El nombre del arreglo
- El número de dimensiones
- El rango (límites) de cada dimensión

Sintáxis:

Dimension <Nombre_Arreglo> (Rango…)….

Tipo Nombre_Arreglo (Rango…)…

Nombre_Arreglo: Nombre del arreglo
Rango: Especifica el límite inferior y superior para la dimensión. Se debe usar un rango para cada dimensión, se puede usar hasta 7 (siete) dimensiones.
Ejemplo1:
DIMENSION *larreglo* (1:2,0:2)

Donde: *larreglo* es un arreglo de 2 filas (1 y 2) y tres columnas (0,1 y 2)

1. El límite inferior: Debe ser una constante entera, puede ser negativa, cero o positiva. Por defecto su valor es **1**
2. Límite superior, constante entera, puede ser positiva, cero o negativa. Debe ser mayor o igual al límite inferior.

Ejemplo2:
DIMENSION larreglo(2,0:2)
Es la misma declaración del ejemplo1

Ejemplos de declaración de Arrays:

1. REAL A(10,2,3) ¡Declara arreglo real
2. DIMENSION A(10,2,3) ¡Declara un arreglo
3. INTEGER M(10,10,10), K(-3:6,4:13,0:9) ¡Declara arreglos M,K entero
4. REAL IMAN(10),B(10),AA(3,3) ¡Declara tres arreglos reales

Ejemplo3
REAL A(3,3),B(3,3),C(89),R
B(2,2) = 4.5
R=7.0
C(INT(R)*2+1)=2.0 **¡Elemento 15 de C = 2.0**
A(1,2)=B(INT(C(15)),INT(SQRT(R)))
!Elemento A(1,2) = Elemento B(2,2) = 4.5

Ejemplo4
Plantear el pseudo código para sumar dos matrices **nXm**

```
      Dimension Nmat(1:10,1:20), Mmat(1:10,1:20), Nresult(1:10,1:20)
      Read(*,*) n,m
      Do 10 I = 1,n
      Do 10 J = 1,m
10    Nresult(I,j)=Nmat(I,J)+Nmat(I,J)
      End do
      End do
```

Expresiones Escalares:

Las expresiones pueden ser escalares o de arreglos. Son escalares si usan variables simples (no declaradas como arreglos)
Ejemplo5
Q + 2.3 + R, donde Q y R son variables simples.

Expresiones Arreglo:
Si tenemos:
DIMENSION A(10), B(10
Entonces:
A+B, C*D, son expresiones arreglo para sumar, multiplicar dos arreglos, siendo que ambos deben ser compatibles.

Si un operando es escalar, es tratado como un arreglo compatible, con cada uno de sus elementos igual al valor escalar: Ejemplo sí: A+r

En este ejemplo si **r** es escalra su valor es sumado a cada elemento del arreglo A.

DECLARACION - Where

Permite ejecutar asignaciones de forma selectiva sobre los elementos de un arreglo
.WHERE (Condición)
Sentencia-asignación
[ELSEWHERE]
Sentencia-asignación
END

- Solo se usan sentencias de asignación.
- Para cada elemento del arreglo se verifica la condición:
  a. Si se cumple se ejecuta la asignación que sigue al Where
  b. Sino se cumple se ejecuta la asignación que sigue al Elsewhere (si existe)

Ejemplo6:
El siguiente ejemplo usa dos métodos diferentes para obtener el cuadrado de todos los elementos que son positivos:
Método 1: Usando la sentencia IF

```
REAL, DIMENSION(10)::A
DO I =1,10
  READ*,A(I)
END DO
DO I = 1,10
 IF (A(I) > 0)  A(I)=A(I)**2
END DO
```

Método 2: Usando la sentencia WHERE

```
REAL, DIMENSION(10):: A
DO I = 1,10
        READ*,A(I)
END DO
WHERE (A>0) A=A**2
```

```
 2
-2
 2
-2
 2
-2
 2
 3
 2
-2
     4.000000      -2.000000       4.000000      -2.000000
     4.000000      -2.000000       4.000000       9.000000
     4.000000      -2.000000
Press any key to continue
```

Ejemplo7:

```
INTEGER, DIMENSION(100):: IN,OUT
```

```
1
2
15
16
2
16
13
15
18
20
        0
        0
```

```
DO I = 1,100
        READ*, IN(I)
END DO
WHERE (IN<8)
        OUT=0
ELSEWHERE
        OUT=15
END WHERE
DO I = 1,100
PRINT*, OUT(I)
END DO
END
```

LECTURA DE ARREGLOS

```
Dimension A(1:20), B(1:2,1:5)
Read(*,*) A(1) ¡Lee una posición del arreglo
Read(*,*) A  ¡Lee todo el arreglo
Read(*,*) B
¡Lee el arreglo por columnas: 1,1;2,1;2,1; etc.
```

- Se puede usar un Do implicto en el Read: ejemplo

```
INTEGER Mat(2,2)
READ(*,*) ((Mat(J,K), K=1,2), J=1,2)
WRITE(*,*) Mat(1,1);WRITE(*,*) Mat(1,2)
WRITE(*,*) Mat(2,1);WRITE(*,*) Mat(2,2)
!write(*,*) ((Mat(J,K), K=1,2), J=1,2)
END
```

```
3 4 5 6
        3
        4
        5
        6
Press any key to continue
```

PROBLEMAS DE ARREGLOS:

1. Escribir un programa que recibe 15 valores reales positivos y los almacena en un vector. Determinar los valores mayor y menor, Imprimir el vector entero y los dos valores computados. Use la sentencia DO implicito en cualquier sentencia de entrada o salida

```
      REAL A(15),MAX,MIN
      READ*, (A(I),I=1,15)
      MAX=A(1)
      MIN=A(1)
      DO 10 I = 2,15
      IF (MAX < A(I)) MAX = A(I)
      IF (MIN > A(I)) MIN = A(I)
10    CONTINUE
      PRINT*, (A(I),I=1,4)
      PRINT*, (A(I),I=5,8)
      PRINT*, (A(I),I=9,12)
      PRINT*, (A(I),I=13,15)
      PRINT*
      PRINT*,MAX,MIN
      STOP
      END
```

```
Datos de entrada:
11.45  13.51  17.92  18.45
13.12  18.39  24.05  13.27
14.79  23.91  17.81  10.74
20.06  13.83  11.67
```

```
 11.450000     13.510000     17.920000     18.450000
 13.120000     18.390000     24.050000     13.270000
 14.790000     23.910000     17.810000     10.740000
 20.060000     13.830000     11.670000

 24.050000     10.740000
```

2. Escribase un programa para desarrollar las siguientes operaciones de matrices. Donde sea necesario, leer los datos de las tarjetas y comprobar las dimensiones para asegurarse de que son apropiados antes de comenzar los cálculos. Los datos constan de las dimensiones de la matriz y los valores de sus elementos

Sumar dos matrices:

Ing. William Chauca Nolasco
96933582

```
      !Suma de matrices M*N
      REAL A(100,100),B(100,100),C(100,100)
      READ*,M,N,((A(I,J),J=1,N),I=1,M)
      READ*,K,L,((B(I,J),J=1,L),I=1,K)
      IF (M .NE. K) GO TO 30
      IF (N .NE. L) GO TO 30
      DO 10 I=1,M
      DO 10 J=1,N
10    C(I,J)=A(I,J)+B(I,J)
      DO 20 I = 1,M
20    PRINT*, (C(I,J),J=1,N)
30    STOP
      END
```

Datos de Prueba:

$$A=\begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad B=\begin{bmatrix} 1 & -2 & 1 & 0 \\ 1 & -2 & 2 & -3 \\ 0 & 1 & -1 & 1 \\ -2 & 3 & -2 & 3 \end{bmatrix}$$

```
4 4
1.0 2.0 3.0 1.0
1.0 3.0 3.0 2.0
2.0 4.0 3.0 3.0
1.0 1.0 1.0 1.0

4 4
1.0   -2.0  1.0   0.0
1.0   -2.0  2.0  -3.0
0.0    1.0 -1.0   1.0
-2.0  3.0  -2.0   3.0
```

```
     2.000000    0.000000E+00    4.000000     1.000000
     2.000000     1.000000       5.000000    -1.000000
     2.000000     5.000000       2.000000     4.000000
    -1.000000     4.000000      -1.000000     4.000000
Stop - Program terminated.
Press any key to continue
```

Multiplicar dos matrices:

```
      !Multiplicacion de matrices
      REAL A(100,100),B(100,100),C(100,100)
      READ*,M,L1,((A(I,J),J=1,L1),I=1,M)
      READ*,L2,N,((B(I,J),J=1,N),I=1,L2)
      IF (L1 .NE. L2) GO TO 40
      DO 20 I =1,M
      DO 20 J=1,N
      SUMA=0.0
      DO 10 K=1,L1
10    SUMA = SUMA+A(I,K)*B(K,J)
20    C(I,J) = SUMA
      DO 30 I = 1,M
30    PRINT*,(C(I,J),J=1,N)
40    STOP
      END
```

Los datos de prueba son los mismos del ejemplo anterior:

```
4 4
1.0 2.0 3.0 1.0
1.0 3.0 3.0 2.0
2.0 4.0 3.0 3.0
1.0 1.0 1.0 1.0
4 4
1.0 -2.0 1.0 0.0
1.0 -2.0 2.0 -3.0
0.0 1.0 -1.0 1.0
-2.0 3.0 -2.0 3.0
  1.000000          0.000000E+00    0.000000E+00    0.000000E+00
  0.000000E+00      1.000000        0.000000E+00    0.000000E+00
  0.000000E+00      0.000000E+00    1.000000        0.000000E+00
  0.000000E+00      0.000000E+00    0.000000E+00    1.000000
Stop - Program terminated.

Press any key to continue
```

INVERTIR UNA MATRIZ:

El método que se ha utilizado para obtener la inversa de una matriz se conoce como "Inversa de las matrices elementales " que se puede consultar en el Libro "MATRICES" por Frank Ayres, Capitulo 7 – Colección SHAUM

```
      !Inversión de matrices
      REAL A(100,100),B(100,100)
      READ*,N,((A(I,J),J=1,N),I=1,N)
      DO 20 I=1,N
      DO 10 J=1,N
10    B(I,J)=0.0
20    B(I,I)=1.0
      DO 80 K=1,N
      C=A(K,K)
      IF (C == 0.0) PRINT*,K,K
      IF (C == 0.0) STOP
      DO 30 J=K,N
30    A(K,J)=A(K,J)/C
      DO 40 J=1,K
40    B(K,J)=B(K,J)/C
      DO 70 I=1,N
      IF (I == K) GOTO 70
      C=A(I,K)
      DO 50 J=K,N
50    A(I,J)=A(I,J)-C*A(K,J)
      DO 60 J=1,K
60    B(I,J)=B(I,J)-C*B(K,J)
70    CONTINUE
80    CONTINUE
      DO 90 I=1,N
90    PRINT*,(B(I,J),J=1,N)
      END
```

```
4
1.0 2.0 3.0 1.0
1.0 3.0 3.0 2.0
2.0 4.0 3.0 3.0
1.0 1.0 1.0 1.0
      1.000000        -2.000000       1.000000      -8.940697E-08
      1.000000        -2.000000       2.000000      -3.000000
     0.000000E+00      1.000000       -1.000000       1.000000
     -2.000000         3.000000       -2.000000       3.000000
Press any key to continue
```

PROBLEMA: Computar el elemento mayor de una matriz:

```
      !Cálculo del elemento mayor de una matriz
      Real A(100,100),MAYOR
      READ*,M,N,((A(I,J),J=1,N),I=1,M)
      MAYOR = A(1,1)
      DO 20 I=1,M
      DO 10 J=1,N
      IF (MAYOR .GE. A(I,J)) GOTO 10
      MAYOR=A(I,J)
10    CONTINUE
20    CONTINUE
      PRINT*,MAYOR
      STOP
      END
```

```
4 4
1.0 2.0 3.0 1.0
1.0 3.0 3.0 2.0
2.0 4.0 3.0 3.0
1.0 1.0 1.0 1.0
     4.000000
Stop - Program terminated.
Press any key to continue
```

PROBLEMA:Inicializar una matriz cuadrada como matriz de identidad.

```
      !Inicializacion de una matriz cuadrada
      ¡como matriz de identidad
      REAL A(100,100)
      READ*,N
      DO 20 I=1,N
      DO 10 J=1,N
10    A(I,J)=0.0
20    A(I,I)=1.0
      DO 30 I=1,N
30    PRINT*,(A(I,J),J=1,N)
```

```
4
  1.000000        0.000000E+00   0.000000E+00   0.000000E+00
  0.000000E+00    1.000000       0.000000E+00   0.000000E+00
  0.000000E+00    0.000000E+00   1.000000       0.000000E+00
  0.000000E+00    0.000000E+00   0.000000E+00   1.000000
Stop - Program terminated.
Press any key to continue
```

```
      STOP
      END
```

PROBLEMA: Inicializar cualquier matriz de dimensiones dadas para que sea una matriz cero.

```
      !Inicialización de una matriz como matriz cero
      Real A(100,100)
      Read*, M,N
      DO 10 I=1,M
      DO 10 J=1,N
10    A(I,J)=0.0
      DO 20 I=1,M
20    PRINT*,(A(I,J),J=1,N)
      STOP
      END
```

```
 4 4
  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
Stop - Program terminated.
Press any key to continue
```

PROBLEMA:
Hállese la suma de los cuadrados de los elementos diagonales de cualquier matriz cuadrada.

```
      !Suma de los cuadrado de los elementos
      !diagonales de cualquier matriz cuadrada
      REAL A(100,100)
      READ*,N,((A(I,J),J=1,N),I=1,N)
      SUMA=0.0
      DO 10 I=1,N
10    SUMA=SUMA+A(I,I)**2
      PRINT*,SUMA
      STOP
      END
```

```
 4
 1.0  2.0  3.0  1.0
 1.0  3.0  3.0  2.0
 2.0  4.0  3.0  3.0
 1.0  1.0  1.0  1.0
     20.000000
Stop - Program terminated.
Press any key to continue
```

PROBLEMA:
Escribase un programa que prepare e inicialice un vector X de 10 elementos, de tal manera que cada elemento contenga el mismo valor real, o sea, 3.456789. Entonces imprimir este vector usando la sentencia:
PRINT 6, (X(I),I=1,10), para cada uno de los siguientes formatos:

1. 6 FORMAT (‘O’,6F10.6)
2. 6 FORMAT(‘O’,F10.5,F10.4,F10.3,F10.2,F10.1,F10.0)
3. 6 FORMAT(‘ ‘,6F10.8)
4. 6 FORMAT(‘ ‘,6F5.4)
5. 6 FORMAT(‘ ‘,F15.3,6X,F15.2)
6. 6 FORMAT(‘O’,4E15.7)
7. 6 FORMAT(‘O’,E15.6,E15.5,E15.4,E15.3,E15.2)
8. 6 FORMAT’O‘,5E12.8)
9. 6 FORMAT(‘O’,E17.4,F17.4)

Será muy instructivo para el lector predecir cuáles serán los resultados antes de que el programa sea corrido en la computadora:

```
      Real X(10)
      Do 10 I =1,10
10    X(I)=3.456789
      Print 1,(X(I),I =1,10)
1     Format('O',6F10.6)
      Print 2,(X(I),I =1,10)
2     Format('O',F10.5,F10.4,F10.3,F10.2,F10.1,F10.0)
      Print 3,(X(I),I =1,10)
3     Format('',6F10.8)
      Print 4,(X(I),I =1,10)
4     Format('',6F5.4)
      Print 5,(X(I),I =1,10)
5     Format('',F15.3,6X,F15.2)
      Print 6,(X(I),I =1,10)
6     Format('O',4E15.7)
      Print 7,(X(I),I =1,10)

7     Format('O',E15.6,E15.5,E15.4,E15.3,E15.2)
      Print 8,(X(I),I =1,10)
8     Format('O',5E12.8)
      Print 9,(X(I),I =1,10)
```

```
  3.456789  3.456789  3.456789  3.456789  3.456789  3.456789
  3.456789  3.456789  3.456789  3.456789
   3.45679   3.4568    3.457     3.46      3.5       3.
   3.45679   3.4568    3.457     3.46
3.456789003.456789003.456789003.456789003.456789003.45678900
3.456789003.456789003.456789003.45678900
****************************
********************
      3.457             3.46
      3.457             3.46
      3.457             3.46
      3.457             3.46
```

```
9       Format('O',E17.4,F17.4)
        Stop
        End
```

Nota: Cuando se ejecutan las sentencias: Print 4, Print 9
Como no se ha indicado el suficiente espacio en las sentencias FORMAT correspondientes, aparecen asteriscos en la respuesta dada por el computador.

PROBLEMA:
Escríbase un programa que prepare e inicialice un vector **I** de 10 elementos, de tal manera que cada elemento contenga el mismo valor entero, o sea, 123456. Imprimir entonces el vector usando la sentencia: PRINT 8, (I(J),J=1,10)
Con cada una de las siguientes sentencias FORMAT

1. 8 FORMAT('O',6I10)
2. 8 FORMAT('',I10,I9,I8,I7,I6,I5)
3. 8 FORMAT('',10I6)
4. 8 FORMAT('O',3I8,3X,2I8)
5. 8 FORMAT('',I20)
6. 8 FORMAT('',10F13.6)

```
        INTEGER I(10)
        DO 10 J=1,10
10      I(J) = 123456
        PRINT 1,(I(J),J=1,10)
1       FORMAT('O',6I10)
        PRINT 2,(I(J),J=1,10)
2       FORMAT('',I10,I9,I8,I7,I6,I5)
        PRINT 3, (I(J),J=1,10)
3       FORMAT('',10I6)
        PRINT 4, (I(J),J=1,10)
4       FORMAT('O',3I8,3X,2I8)
        PRINT 5, (I(J),J=1,10)
5       FORMAT('',I20)
        PRINT 6, (I(J),J=1,10)
6       FORMAT('',10F13.6)
        STOP
        END
```

```
  123456   123456   123456   123456   123456   123456
  123456   123456   123456   123456
  123456   123456   123456   123456   123456   *****
  123456   123456   123456   123456
1234561234561234561234561234561234561234561234561234561234561234
  123456  123456  123456    123456  123456
  123456  123456  123456    123456  123456
          123456
          123456
          123456
          123456
          123456
          123456
          123456
          123456
          123456
          123456
run-time error F6207: WRITE(CON)
- I edit descriptor expected for INTEGER
Press any key to continue
```

Nótese que las sentencias: PRINT 6, (I(J),J=1,10)
6 FORMAT('',10F13.6)
Da lugar a que el programa termine indicando error de formato, a una variable entera no se le puede imprimir con formato de variable real.

Ing. William Chauca Nolasco
96933582

PROBLEMA: Computar el producto interno de dos cualesquiera de los vectores:

```
      !Producto interno de dos vectores
      Real A(100),B(100)
      READ*,N
      DO 20 K=1,N
      READ*,L,(A(I),I=1,L)
      READ*,M,(B(I),I=1,M)
      IF(L .GT. M) L=M
      SUMA=0.0
      DO 10 I=1,L
10    SUMA=SUMA+A(I)*B(I)
20    PRINT*,SUMA
      STOP
      END
```

```
 2
 3 2.0   2.0  -1.0
 3 6.0  -3.0   2.0
       4.000000
 3 3.0   4.0   12.0
 2 6.0   8.0
      50.000000
Stop - Program terminated.
Press any key to continue
```

Nota: En el programa **N** significa el número de veces que se quiere ejecutar el programa, Observe que no es necesario que los vectores tengan las mismas dimensiones.

PROBLEMA:Computar el ángulo entre dos vectores cualesquiera:

```
      !Angulo entre dos vectores
      REAL A(100),B(100),LA,LB
      READ*,N
      DO 40 K=1,N
      READ*,L,(A(I),I=1,L)
      READ*,M,(B(I),I=1,M)
      LA=0.0
      DO 10 I=1,L
10    LA=LA+A(I)**2
      LB=0.0
      DO 20 I=1,M
20    LB=LB+B(I)**2
      IF (L .GT. M) L=M
      SUMA=0.0
      DO 30 I=1,L
30    SUMA=SUMA+A(I)*B(I)
      ANGULO=SUMA/SQRT(LA*LB)
40    PRINT*,ANGULO
      STOP
      END
```

```
 2
 3 2.0    2.0  -1.0
 3 6.0  -3.0    2.0
   1.904762E-01
 3 3.0   4.0   12.0
 2 6.0   8.0
   3.846154E-01
Stop - Program terminated.
```

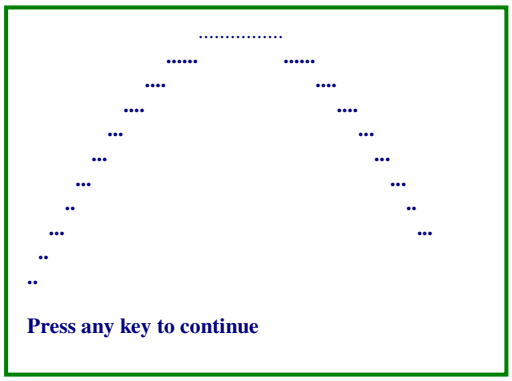
En el programa **N** significa el número de veces que se quiere ejecutar el programa, no es necesario que los vectores tengan las mismas dimensiones.

GENERACION DE GRAFICOS EN FORTRAN

La mayoría de las pantallas del monitor de las microcomputadoras miden 80 columnas por 20 filas. Nosotros usaremos un conjunto de carácteres de este tamaño para dibujar sobre ella una funcion. El proceso involucra tramando 80 puntos desde **-Pi** a **+Pi** y escalando estos puntos para adaptarlo dentro de los límites del conjunto.

Ejemplo:

```
      Program Gráfico
      INTEGER COLS,ROWS
      PARAMETER (COLS=80, ROWS=20, PI=3.1416)
      CHARACTER*1 SCREEN(ROWS,COLS)
      DATA SCREEN /1600*"/
      DO JX=1,COLS
      X=(-PI+(JX-1)*(PI))/((COLS-1)-2)
      Y=SIN(X)
      IY = 1+NINT((Y-1.0)/(-2.0)*(ROWS-1))
      SCREEN(IY,JX) = '.'
      END DO
      DO I=1,ROWS-1
      PRINT*,(SCREEN(I,J),J=1,COLS-4)
      END DO
      END PROGRAM Grafico
```


Press any key to continue