Programación Avanzada

FORTRAN®

Semestre 2019-1

Ing. Juan Carlos Sabido Alcántara

Ingeniero Petrolero

Facultad de Ingeniería UNAM

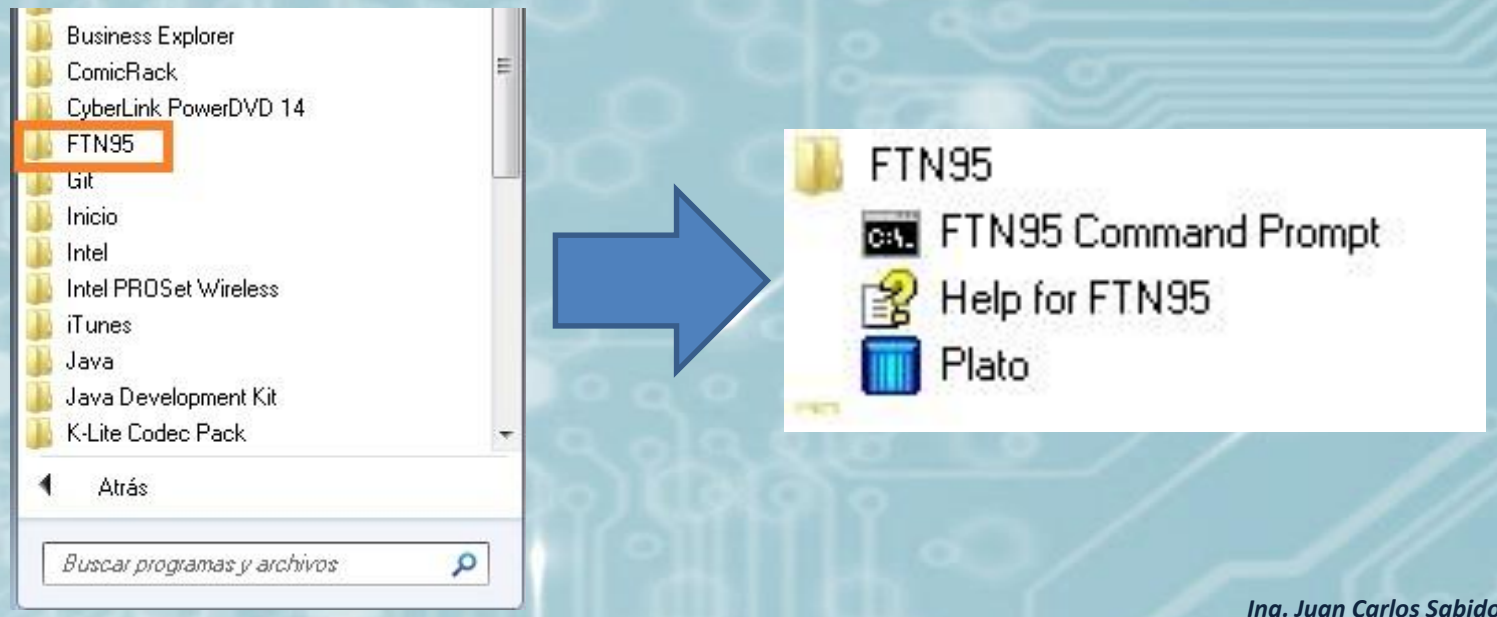

4G
5:15
777-777-7777
CALLING

FORTRAN®

- Para efectos de este curso utilizaremos la versión de FORTRAN® que provee de manera gratuita el sitio:

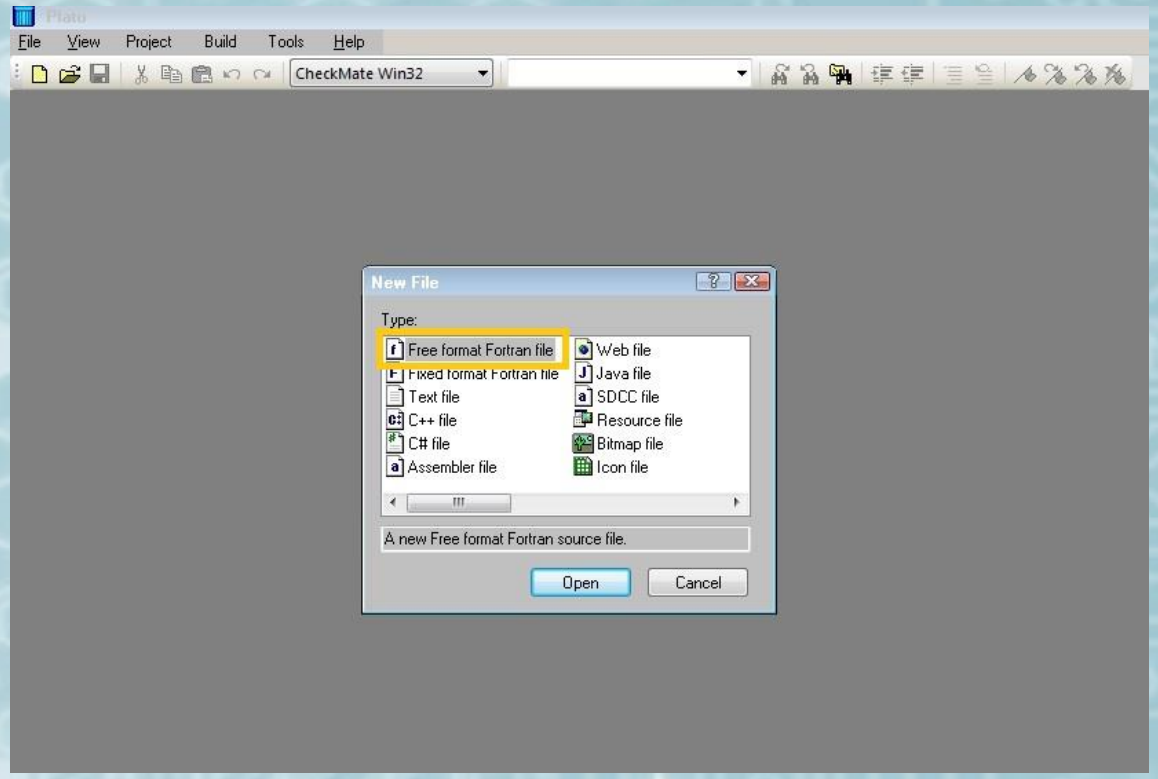
  http://www.silverfrost.com/default.aspx

- Una vez instalado en el equipo lo podemos encontrar en “Todos los Programas” en la carpeta “FTN95”, en donde aparecerá el programa “PLATO”:



FORTRAN®

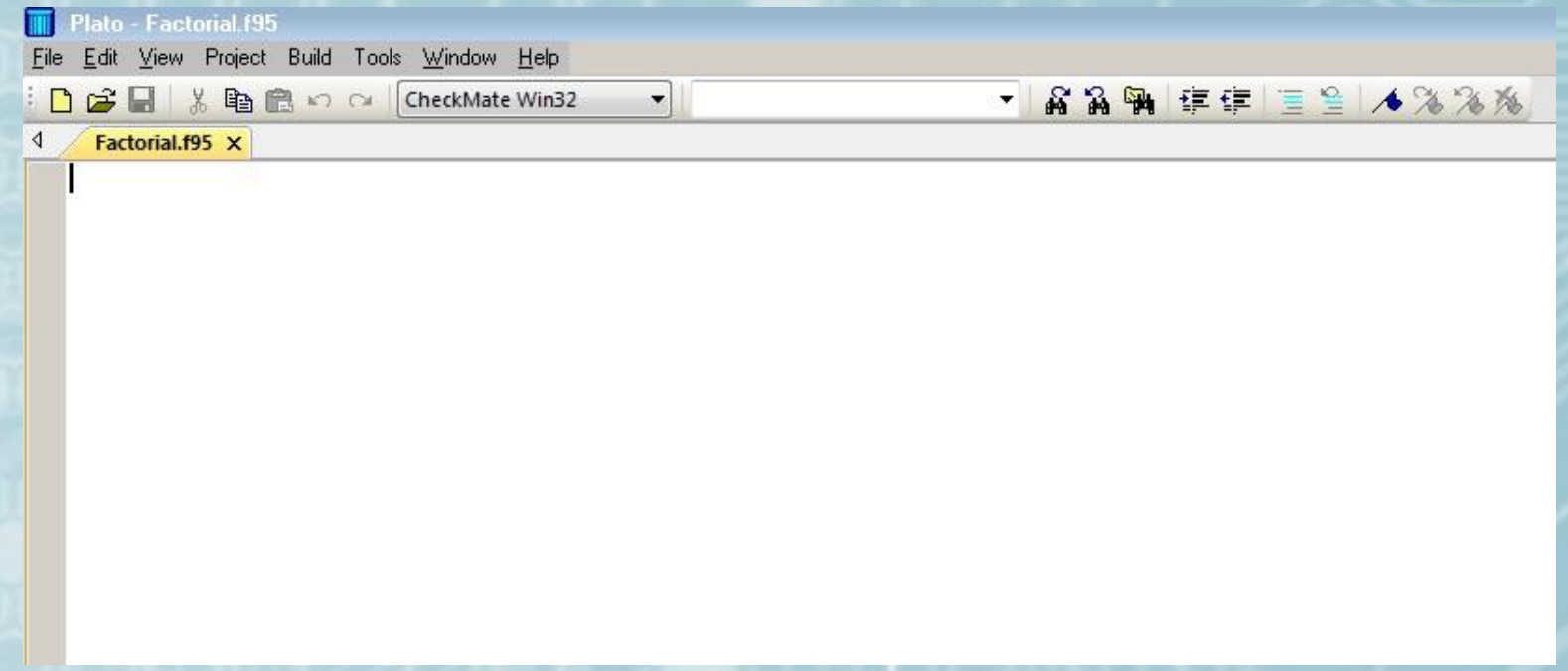
- Cuando entramos al lenguaje, basta con ir al menú “FILE”, elegir la opción “NEW”, y después seleccionar “Free format FORTRAN® file” y aparecerá el espacio en donde podemos comenzar a programar.



Ing. Juan Carlos Sabido Alcántara

FORTRAN®

- Antes de iniciar a escribir el código la recomendación es guardar el archivo en una carpeta, ya que el lenguaje generará varios elementos al momento de compilar el programa, esto lo hace en el lugar en donde esté ubicado el archivo que contiene el código, por lo que agruparlo en una carpeta exclusiva del programa es lo ideal.



FORTRAN®

- ***Estructura de un programa en FORTRAN®.***
- ***Sección de declaración.***
- El tipo de variable debe de ser declarada en la sección de declaración al comienzo del programa, después de la instrucción no ejecutable **PROGRAM.** La instrucción **IMPLICIT NONE** deshabilita la definición implícita de las variables. Cuando se emplea **IMPLICIT NONE** se debe de declarar explícitamente todas las variables utilizadas dentro del programa, aunque se puede omitir, es recomendable utilizarla pues esto permite identificar rápidamente posibles fuentes de error durante la compilación del programa e incluso durante la ejecución de este. La instrucción **IMPLICIT NONE** debe aparecer después de la instrucción **PROGRAM** y antes de cualquier instrucción de declaración.

FORTRAN®

- ***Estructura de un programa en FORTRAN®.***
- ***Sección de ejecución.***
- La sección de ejecución consiste de una o más instrucciones ejecutables que describe las acciones que va a realizar el programa.
- Para esto se utilizan algunas instrucciones definidas en FORTRAN®, además se pueden escribir operaciones aritméticas que permiten calcular, modificar o comparar los valores asumidos por las diferentes variables.
- Algunas instrucciones definidas para FORTRAN® que se utilizan en esta sección son:
  - **PRINT ,*** permite escribir en pantalla algún tipo de mensaje.
  - **WRITE (*,*)** se utiliza para escribir en pantalla o en archivos externos del programa.
  - **READ (*,*)** se utiliza para asignar valores a las variables, estos valores se pueden leer directamente de la información que introduce el usuario en pantalla o de algún archivo externo al programa.

FORTRAN®

- ***Estructura de un programa en FORTRAN®.***
- ***Sección de terminación.***
- La sección de terminación consiste de las instrucciones **STOP** y **END PROGRAM,** la instrucción **STOP** detiene el programa, mientras que la instrucción **END PROGRAM** indica que no hay más instrucciones a ejecutar.
- En algunos programas que utilizan subrutinas estas se colocan después del **END PROGRAM**, esto se debe a que las subrutinas solo pueden ser utilizadas por y en el programa principal, por lo que a lo largo de este en algún punto especifico se mandan a llamar estas subrutinas, por esto las subrutinas pueden ser ubicadas después de la línea que contiene el **END PROGRAM**, además se pueden crear módulos en donde se almacenen las subrutinas que utilice el programa.

FORTRAN®

- ***Compilación y ejecución de un programa.***
- Una vez que se escribió el código, es necesario compilarlo para así poder ejecutar el programa. Al compilar el código fuente lo que ocurre es un análisis orientado a encontrar errores, es decir, se verifica que todas las líneas contengan instrucciones compatibles con el lenguaje, se detectan variables no declaradas en la sección correspondiente y que sin embargo se encuentran a lo largo del código, etc. En el caso de ***FTN95® (PLATO®)*** se especifican las líneas del código en las que se encuentran estos errores.
- Cabe aclarar que existen errores que los compiladores no pueden detectar, ya que no están asociados a la sintaxis del código, si no que pueden deberse a una interpretación errónea que da el programador a la solución del problema, estos generalmente ocurren durante la ejecución del programa, y pueden deberse a operaciones inválidas como una división entre cero, exceder el orden de los arreglos, etc., con la práctica y la experiencia es mas sencillo detectar los errores que indica el compilador o los que se generan por una mala interpretación.

FORTRAN®

- ***Compilación y ejecución de un programa.***
- Para compilar el código solo se necesita presionar ***Ctrl+F7*** o ir al menú ***"Build"*** dar click en la opción ***"Compile"***  . En la parte inferior de la ventana aparece otra conocida como ***output*** en donde se muestran las líneas que tienen algún error y de que tipo se trata, la última línea muestra la cantidad de errores y de advertencias, la diferencia entre estos puntos es que si existe por lo menos un error el programa no se podrá ejecutar, mientras que no importa cuantas advertencias o comentarios existan, el programa puede ser ejecutado, sin embargo es muy probable que el programa se desborde, razón por la que se recomienda depurar el código hasta obtener cero errores y cero advertencias.

```
Output
Compiling file: Machine.f95
G:\VAIO 31 07 2016\Clase Programaci▯n Avanzada\Programas\Machine\Machine.F95(4) : error 774 - T
G:\VAIO 31 07 2016\Clase Programaci▯n Avanzada\Programas\Machine\Machine.F95(8) : comment 368 -
Compilation failed.
```

FORTRAN®

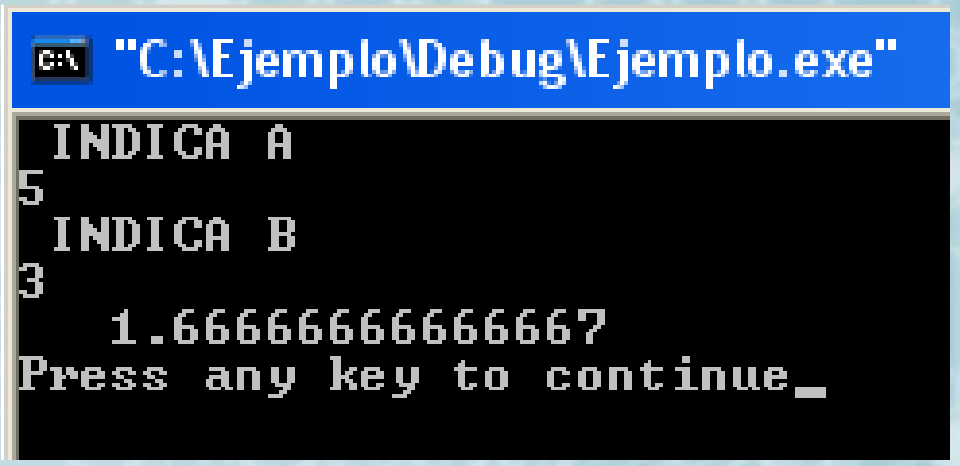
- ***Compilación y ejecución de un programa.***
- Si después de compilar el código no se tienen errores ni advertencias se puede ejecutar el programa, para esto basta con presionar ***Ctrl+F5*** o ir al menú ***"Build"*** y hacer click en la opción ***"Start Run"***, esto generara en la carpeta del programa un archivo ejecutable con la extensión exe y ejecutara el programa. El programa se ejecuta en la ventana de Símbolo del Sistema antes conocido como MsDOS.

FORTRAN®

- ***Compilación y ejecución de un programa.***
- ***Ejemplo*** sencillo de la estructura de un programa escrito en FORTRAN®:
  - *PROGRAM PRIMERO*
  - *IMPLICIT NONE*
  - *REAL(KIND=2)::A,B,C*
  - *WRITE(*,*)'INDICA A'*
  - *READ(*,*)A*
  - *WRITE(*,*)'INDICA B'*
  - *READ(*,*)B*
  - *C=A/B*
  - *WRITE(*,*)C*
  - *END PROGRAM*
- Las líneas que contienen la instrucción **WRITE(*,*)** sirven para mostrar en pantalla los mensajes que están entre apóstrofes (‘), también se pueden utilizar comillas (“), las líneas **READ(*,*)** asignan el valor introducido en pantalla por el usuario a las variables que ahí se indican.

FORTRAN®

- ***Compilación y ejecución de un programa.***
- Una vez que se asignaron valores a las variables A y B se asigna el valor a C resultante de la división A entre B mediante **C=A/B** para después mandarlo a imprimir a pantalla. El programa en ejecución es el que se muestra en la imagen.



- *PROGRAM PRIMERO*
- *IMPLICIT NONE*
- *REAL(KIND=2)::A,B,C*
- *WRITE(*,*)'INDICA A'*
- *READ(*,*)A*
- *WRITE(*,*)'INDICA B'*
- *READ(*,*)B*
- *C=A/B*
- *WRITE(*,*)C*
- *END PROGRAM*

FORTRAN®

- ***Variables.***
- Para FORTRAN® (y para cualquier lenguaje) existen diferentes tipos de variables con las que se construye el programa, en este caso se tienen las siguientes:
  - **INTEGER:** Variable numérica de tipo entero.
  - **REAL:** Variable numérica de tipo real.
  - **CHARACTER:** Variable de tipo carácter.
  - **LOGICAL:** Variables lógicas.
- Las variables pueden adquirir como valores una combinación de dígitos, letras, o ambos dependiendo del tipo de variable, para nombrarlas se pueden utilizar nombres que contengan hasta 31 caracteres de longitud, aunque esto no es recomendable, pues resulta impráctico, además el primer carácter del nombre de la variable no puede ser del tipo numérico.

FORTRAN®

- ***Variables.***
- Por ejemplo para dar nombre a una variable referente a densidad relativa del aceite, *denrelo* sería correcto, pues representa adecuadamente y de forma sencilla el concepto con el que se asocia la variable a lo largo del programa y con las operaciones que la involucran, en cambio *densidad_relativa_aceite* no es práctico, pues aunque presenta claramente el concepto con el que esta asociada es muy largo.
- Para declarar una variable se deben escribir las siguientes líneas para cada tipo de variables:
  - *INTEGER::I,J,K*
  - *REAL(KIND=2)::A,B,C*
  - *REAL::X,Y*
  - *CHARACTER(50)::RUTA*
  - *LOGICAL::COMPROBAR*

FORTRAN®

- ***Variables.***
- Las variables I, J, K son de tipo entero, las variables de A, B, C son de tipo real de doble precisión, esto lo define el contenido del paréntesis (KIND=2), ***(recordando que estamos usando FTN95 ® (PLATO ®), en otras plataformas que manejen FORTRAN® puede cambiar la instrucción que defina como doble precisión a una variable REAL)***, en cambio las variables X, Y son de tipo real pero de precisión sencilla, esto se refiere al modo en que la computadora representa los números visto anteriormente, la variable RUTA es de tipo carácter y puede tener una longitud de cincuenta caracteres, esto lo define el número 50 escrito entre paréntesis, la variable COMPROBAR es de tipo lógica.
  - *INTEGER::I,J,K*
  - *REAL(KIND=2)::A,B,C*
  - *REAL::X,Y*
  - *CHARACTER(50)::RUTA*
  - *LOGICAL::COMPROBAR*

FORTRAN®

- ***Operadores lógicos, aritméticos y de comparación.***
- **Operadores lógicos.**
- Los operadores con los que trabaja FORTRAN® ® son los siguientes:

Operador	*Significado*	*Definición*
A.AND.B	Y	Resultado True si A y B son verdaderas
A.OR.B	O	resultado true si A o B es verdadera
A.EQV.B	EQUIVALENCIA	resultado true si A=B
A.NEQV.B	NO EQUIVALENCIA	
A.NOT.B	NO EQUIVALENCIA	

- **Operadores aritméticos.**
- Los operadores con los que trabaja FORTRAN® ® son los siguientes:

Operador	*Significado*
+	Suma
-	Resta
*	Multiplicación
/	División
**	Exponenciación

Ing. Juan Carlos Sabido Alcántara

FORTRAN®

- ***Operadores lógicos, aritméticos y de comparación.***
- **Operadores de comparación.**
- Los operadores con los que trabaja FORTRAN® son los siguientes:

Operador		Significado
Estilo1	Estilo2	
.EQ.	==	igual
.NE.	/=	diferente
.LT.	<	menor que
.LE.	<=	Menor o igual que
.GT.	>	mayor que
.GE.	>=	Mayor o igual que

- ***Jerarquía operacional.***
- La jerarquía que sigue FORTRAN® y la mayoría de los lenguajes para evaluar las operaciones aritméticas que componen el programa, comienza por evaluar el contenido de los paréntesis que forman una expresión desde adentro hacia fuera, en caso de contener exponentes estos se evalúan de derecha a izquierda, las sumas, restas, multiplicaciones y divisiones se evalúan de izquierda a derecha.

FORTRAN®

- **Estructuras de control.**
- Estas estructuras permiten controlar el orden con el que el programa realizará las operaciones indicadas.
- ***Estructuras de decisión:*** Por medio de condiciones limitadas mediante comparaciones de una variable con un valor patrón, o entre dos o más variables, se indica al programa qué instrucciones debe o no realizar. Para esto existe la estructura **IF … THEN … ELSE,** el código escrito en FORTRAN® es:

IF *(expresiones lógicas, comparativas o ambas)* ***THEN***
instrucciones
instrucciones
ELSE
instrucciones
instrucciones
END IF

FORTRAN®

- **Estructuras de control.**
- ***Estructuras de decisión:*** Si se cumple la condición indicada se realizará el primer bloque de instrucciones y se omitirá el segundo, si no se cumple se omite el primero y se realizan las instrucciones del segundo bloque. Se puede observar que solo se verifica que se cumpla una condición, pero puede darse el caso en que sea necesario verificar más condiciones para decidir que bloque de instrucciones realizar, cuando se presenta este caso existe la siguiente estructura de decisión:

IF *(expresión lógica, comparativa o ambas)* ***THEN***
instrucciones
ELSEIF *(expresión lógica, comparativa o ambas)* ***THEN***
instrucciones
ELSE
instrucciones
END IF

FORTRAN®

- **Estructuras de control.**
- Otra estructura utilizada en FORTRAN® es **SELECT CASE**, esta estructura permite elegir entre dos o mas opciones, la diferencia principal con el **IF … THEN … ELSE**, es que el **SELECT CASE** solo puede hacer uso de un valor especifico para elegir entre una u otra opción, mientras que con **IF** el valor que toma la variable para poder decidir puede estar dentro de un rango de valores definidos por el usuario. La estructura es la siguiente:

```
SELECT CASE (expresión)
CASE(case selector 1)
        instrucciones
        instrucciones
CASE(case selector 2)
        instrucciones
        instrucciones
CASE DEFAULT
        instrucciones
        instrucciones
END SELECT
```

El case DEFAULT es opcional, si se utiliza se ejecutará el bloque de instrucciones que le corresponde solo si el valor de expresión esta fuera del rango de todos los case selector.

FORTRAN®

- **Estructuras de control.**
- ***Estructuras de repetición o ciclos.***
- Permiten ejecutar un grupo de instrucciones durante un número finito de veces, para esto existen dos tipos de sentencia, los ciclos iterativos y los ciclos condicionados.
- La estructura de los ciclos iterativos es la siguiente:

DO *valor inicial, valor final, incremento*

instrucciones

instrucciones

END DO

FORTRAN®

- **Estructuras de control.**
- ***Estructuras de repetición o ciclos.***
- La estructura de los ciclos condicionados puede escribirse de dos formas, la primera es:

DO WHILE *(condición)*
instrucciones
instrucciones
END DO

- Mientras la condición se cumpla se repetirán las instrucciones dentro del ciclo, la segunda manera es:

DO
instrucciones
instrucciones
IF *(expresión lógica, comparativa o ambas)* ***EXIT***
END DO

- Se comienza a ejecutar las instrucciones dentro del ciclo, se realiza una decisión (IF) si se cumple o no la instrucción **EXIT** indica que el ciclo se termina.

FORTRAN®

- **Estructuras de control.**
- ***Ejemplos de estructuras de decisión y repetición.***
- **Estructuras de decisión.**
- ***Ejemplo 1.***

```
IF (A>=3) THEN
        C=(A+B)/(A**B)
        ELSEIF (A<=1) THEN
                C=(A+B)**B
        ELSE
                C=A+B
END IF
```

- ***Ejemplo 2.***

```
IF (C<=2.5) THEN
        F=D+E
        IF (F>=1.5) THEN
                G=F+C
        ELSE
                G=F-C
        END IF
ELSE
        G=F/C
END IF
WRITE(*,*)C
WRITE(*,*)F
WRITE(*,*)G
```

- En este ejemplo se puede observar una estructura IF (F>=1.5) anidada, es decir se encuentra dentro de otra estructura IF y solo se evaluara si la estructura mas arriba se cumple.

FORTRAN®

- **Estructuras de control.**
- ***Ejemplos de estructuras de decisión y repetición.***
- **Estructuras de decisión.**
- Un uso común y sencillo del **SELECT CASE** es la creación de un *"menú de opciones"* dentro de un programa, para esto se utiliza un variable que almacenará el número de la opción deseada por el usuario, el siguiente código muestra un pequeño ejemplo de esto:

```
WRITE(*,*)"*****************************************"
WRITE(*,*)"*                                         *"
WRITE(*,*)"*                                         *"
WRITE(*,*)"* ELIJE UNA OPCIÓN:                       *"
WRITE(*,*)"*                                         *"
WRITE(*,*)"* 1. SUMA                                 *"
WRITE(*,*)"* 2. RESTA                                *"
WRITE(*,*)"* 3. MULTIPLICACIÓN                       *"
WRITE(*,*)"* 4. DIVISIÓN                             *"
WRITE(*,*)"*                                         *"
WRITE(*,*)"*                                         *"
WRITE(*,*)"*****************************************"
READ(*,*)OPCION
```

FORTRAN®

- **Estructuras de control.**

```
SELECT CASE (OPCION)
  CASE(1)
  WRITE(*,*)"*****************************************"
  WRITE(*,*)"*                                       *"
  WRITE(*,*)"*            S U M A                    *"
  WRITE(*,*)"*                                       *"
  WRITE(*,*)"*****************************************"
  CASE(2)
  WRITE(*,*)"*****************************************"
  WRITE(*,*)"*                                       *"
  WRITE(*,*)"*            R E S T A                  *"
  WRITE(*,*)"*                                       *"
  WRITE(*,*)"*****************************************“
  CASE DEFAULT
WRITE(*,*)"*****************************************"
   WRITE(*,*)"*                                      *"
  WRITE(*,*)"*    L A  O P C I O N  N O  E X I S T E  *"
  WRITE(*,*)"*                                      *"
  WRITE(*,*)"*****************************************"
END SELECT
```

- El usuario deberá decidir entre las cuatro opciones a su disposición, por ejemplo, si elige 1 el programa desplegara el mensaje de suma, si elije 2 el de resta, etc., si elije una opción diferente a las cuatro que se le ofrecen el programa salta a la parte del **CASE DEFAULT** e indica que la opción no existe. Aunque este ejemplo es muy sencillo se pueden construir a partir de esto programas más complejos en los que cada opción realice una gran cantidad de operaciones.

FORTRAN®

- **Estructuras de control.**
- **Estructuras de Repetición.**
- ***Ejemplo 1.***

```
SUMA=0
DO I=1,N
        SUMA=SUMA+I
END DO
```

- El ciclo realizará la suma de los números enteros de 1 hasta N, donde N puede ser cualquier valor entero positivo.

- ***Ejemplo 2.***

```
SUMA=0
DO I=1,-N,-1
        SUMA=SUMA + I
END DO
```

- El ciclo realizará la suma de los números enteros de 1 hasta -N, -N puede ser cualquier valor entero negativo, el -1 indica que el ciclo se moverá hacia atrás.

FORTRAN®

- **Ejercicio.**
- Realizar un programa que calcule las siguientes operaciones:

$$y = \sum_{i=1}^{n} \frac{x^2+2}{2} \text{ si } n \geq 10$$

y

$$y = \sum_{i=1}^{n} x \text{ si } 1 \leq n < 10$$

- Realizar la prueba de escritorio para n =3 y construir el programa.

FORTRAN®

- **Arreglos.**
- En cualquier lenguaje de programación se puede hacer uso de variables que reciben el nombre de arreglos, estos toman la forma de vectores y matrices y se pueden utilizar para almacenar valores provenientes de un sistema de ecuaciones lineales o no lineales, o valores provenientes de tablas de datos necesarios para realizar los cálculos requeridos por un programa. FORTRAN® no es la excepción y permite el manejo de vectores, matrices e incluso cubos.

FORTRAN®

- **Arreglos.**
- ***Concepto de vector y matriz en programación.***
- Al indicar a la computadora la existencia de una variable en forma de vector o matriz lo que se está haciendo es reservar un espacio de memoria finito en el que se van a almacenar datos, el espacio reservado es la dimensión del vector o matriz, es decir un vector con una dimensión de 5 significa que es capaz de almacenar cinco datos diferentes, una matriz con dimensión de 3*3 puede almacenar nueve datos. Aunque ese espacio es finito se puede modificar durante la ejecución del programa aumentando o disminuyendo las dimensiones de los arreglos, a esto se le conoce como *“memoria dinámica”.*

FORTRAN®

- **Arreglos.**
- **Declaración de arreglos con dimensión fija.**
- Los arreglos pueden ser variables de tipo entero y real, con dimensión fija o dinámica y se declaran mediante la siguiente línea de código:
  - ***REAL(KIND=2),DIMENSION(5,5)::A***
  - ***REAL(KIND=2),DIMENSION(4)::B***
- Estas líneas indican que la variable A es de tipo real y doble precisión, la sentencia *DIMENSION(5:5)* indica que se trata de un arreglo de tipo matricial con una dimensión de 5*5, es decir, puede almacenar 25 elementos, la siguiente línea indica que la variable B es de tipo real de doble precisión, la sentencia *DIMENSION(4)* indica que se trata de un arreglo en forma de vector con dimensión de 4 elementos.

FORTRAN®

- **Arreglos.**
- **Declaración de arreglos con dimensión variable (memoria dinámica).**
- Si se desea que la dimensión de los arreglos sea variable y se modifique durante la ejecución del programa la declaración se debe realizar de la siguiente forma:
  - ***REAL(KIND=2),ALLOCATABLE,DIMENSION(:,:)::A***
  - ***REAL(KIND=2),ALLOCATABLE,DIMENSION(:)::B***
- Estas líneas indican el tipo de variable y su precisión, adicionalmente la sentencia ALLOCATABLE,DIMENSION(:,:) indica que la variable A es de tipo matricial con dimensión indefinida, esta se asignara durante la ejecución del programa, de igual forma ocurre con la variable B solo que en este caso se trata de un vector.

FORTRAN®

- **Arreglos.**
- **Memoria dinámica.**
- Una vez que se han declarado los arreglos de manera dinámica es necesario indicar al programa durante la ejecución la dimensión que estos tomarán, para estos es necesario hacer uso de una o dos variables, según sea el caso, de tipo entero que definan la dimensión de los arreglos, estas variables enteras deberán leerse durante la ejecución del programa.
- Si los valores enteros que definen la dimensión de los arreglos ya ha sido introducida, la línea de código que permite dimensionar los arreglos es la siguiente:
  - ***ALLOCATE(A(N,M))***
  - ***ALLOCATE(B(N),X(N))***

FORTRAN®

- **Arreglos.**
- **Memoria dinámica.**
- Con esto se indica que a la variable A hay que alojar las variables enteras N y M para definir que esta es un arreglo matricial de dimensión N*M mientras a los vectores B y X hay que alojar la variable N para que su dimensión sea de tamaño N. Si se trata de un programa en el que se puede llegar a utilizar más de una vez los arreglos con diferentes datos y dimensiones es necesario desalojar las variables que dan la dimensión a los arreglos, esto se realiza mediante la siguiente línea:
  - ***DEALLOCATE (A)***
  - ***DEALLOCATE (B,X)***
- Esto significa que las variables A, B, X vuelven a quedar vacías y sin dimensión, por lo que se tendrían que dimensionar una vez más.

FORTRAN®

- **Arreglos**
- ***Lectura e impresión de datos.***
- Para leer los datos que integran un arreglo es necesario las estructuras de repetición, adicionalmente debido a la complejidad y monotonía que puede llegar a tener escribir directamente en la pantalla del programa los datos que conforman el arreglo se vuelve primordial realizar la lectura de datos desde archivos externos al programa, adicionalmente imprimir resultados en un archivo externo permite utilizar estos en otros programas como EXCEL® para obtener gráficas, realizar análisis estadísticos, etc.

FORTRAN®

- **Arreglos.**
- **Lectura en pantalla.**
- Una vez que la dimensión de los arreglos ha que dado establecida, ya sea de forma fija desde la escritura del código del programa o de manera dinámica en el tiempo de ejecución de este, la lectura de los datos de un arreglo en forma de vector se hace de manera sencilla:

```
DO I=1,N
        READ(*,*)B(I)
END DO
```

- Esto significa que se comenzara a leer el valor del elemento B(1), siguiendo con el elemento B(2), el B(3), hasta el N, esto es en caso de ser dinámico, si se trata de un arreglo de dimensión fija la lectura se realizará hasta que se alcance dicha dimensión, en el código solo será necesario escribir en lugar de N el valor fijo de la dimensión del arreglo.

FORTRAN®

- **Arreglos.**
- **Lectura en pantalla.**
- Para leer los datos que formarán un arreglo matricial es un poco más complicado ya que es necesario el uso de un par de ciclos, uno anidado dentro del otro, esto tiene la finalidad de fijar con el primer ciclo los renglones de la matriz y con el segundo moverse sobre los elementos de estos, o en el caso inverso, fijar las columnas y moverse sobre los elementos de estas.

```
DO I=1,N
        DO J=1,M
                READ(*,*)A(I,J)
        END DO
END DO
```

- En este caso se comienza a leer los elementos del primer renglón, es decir, cuando I=1 se leen los elementos A(1,J=1), A(1,J=2), ..., A(1,J=M), cuando I=2 se leen los elementos A(2,J=1), A(2,J=2), ..., A(2,J=M), y así sucesivamente hasta que I=N.

FORTRAN®

- **Arreglos.**
- **Impresión en pantalla.**
- Para realizar la impresión de los valores contenidos en los arreglos vectoriales y matriciales los ciclos *DO* son exactamente los mismos solo es necesario cambiar la instrucción *READ(*,*)* por *WRITE(*,*),* quedando las líneas de código para un vector:

```
DO I=1,N
        WRITE(*,*)B(I)
END DO
```

- Para una matriz:

```
DO I=1,N
        DO J=1,M
                WRITE(*,*)A(I,J)
        END DO
END DO
```

FORTRAN®

- **Arreglos.**
- **Impresión en pantalla.**
- Estas últimas líneas imprimen los elementos de la matriz en forma de columna, si se desea que aparezcan en pantalla como matriz las líneas son:

```
DO I=1,N
        WRITE(*,*)A(I,1:M)
END DO
```

FORTRAN®

- **Arreglos.**
- ***Operaciones con arreglos.***
- Los arreglos pueden utilizarse para hacer cualquier tipo de operación aritmética permitida en FORTRAN®, dependerá del problema que se pretenda resolver y del programador la forma en que estas se realicen. Además los arreglos pueden interactuar con variables sencillas en cualquier momento.
- Cuando interactúan aritméticamente dos variables de tipo arreglo, las operaciones se realizan elemento a elemento, contradiciendo las reglas matemáticas conocidas, es decir, los arreglos aunque tienen formas vectoriales o matriciales no son operadas bajo dichas reglas.

FORTRAN®

- **Arreglos.**
- ***Operaciones con arreglos.***
- **Suma de Matrices en FORTRAN®.**
- La suma de matrices matemáticamente hablando se realiza elemento a elemento, por lo que no existe diferencia con lo que realiza la computadora:

```
READ(*,*)N
ALLOCATE(A(N,N),B(N,N),C(N,N))
        …
        …
        …
DO I=1,N
        DO J=1,N
                        C(I,J)=A(I,J)+B(I,J)
        END DO
END DO
        DO I=1,N
                        WRITE(*,*)C(I,1:N)
        END DO
```

FORTRAN®

- **Arreglos.**
- ***Operaciones con arreglos.***
- **Multiplicación de matrices en FORTRAN®.**
- Según el álgebra la multiplicación de dos matrices A y B debe realizarse multiplicando los elementos A(i,k) por los elementos B(k,j) y sumar el producto de cada una de estas multiplicaciones generando un nuevo elemento C(i,j) de la matriz resultado, es lo que se conoce como renglón por columna. Sin embargo al programar y solamente expresar el producto A(i,j) * B(i,j) se obtendrá una matriz C(i,j) cuyos elementos son muy diferentes de los que se obtendrían respetando el álgebra de matrices.
- **División de matrices FORTRAN®.**
- La división de matrices no esta definida matemáticamente hablando, sin embargo al programarla lo único que ocurre es que se divide el elemento A(i,j) entre su correspondiente B(i,j).

FORTRAN®

- **Arreglos.**
- **Ejercicio.**
- Crear un programa que realice la suma de dos matrices, hacer la prueba de escritorio paso por paso para dos matrices de 3*3.

FORTRAN®

- **Programa #2: Matrices Parte 1.**
- Realizar un programa que trabaje con matrices y vectores utilizando memoria dinámica, y que el usuario pueda elegir entre las opciones realizar lo siguiente:
    1. Producto de un vector por un escalar.
    2. Producto punto entre vectores.
    3. Producto cruz entre vectores.
    4. Producto de una matriz por un escalar.
    5. La transpuesta de una matriz.
- Enviar el código, algoritmo y prueba de escritorio en un archivo de Word al correo de la materia a más tardar a las 6:00 horas del día sábado 2 de marzo de 2018.
- El nombre del archivo debe seguir la siguiente estructura:
    - ***matrices1_primerapellido_segundoapellido_nombre***

FORTRAN®

- **Programa #3: Matrices Parte 2.**
- Completar el programa número tres agregando las siguientes operaciones al menú:
    6. Suma de dos matrices.
    7. Producto de dos matrices.
- Enviar el código, algoritmo y prueba de escritorio en un archivo de Word al correo de la materia a más tardar a las 19:00 horas del día anterior a la siguiente clase.
- El nombre del archivo debe seguir la siguiente estructura:
    - ***matrices2_primerapellido_segundoapellido_nombre***

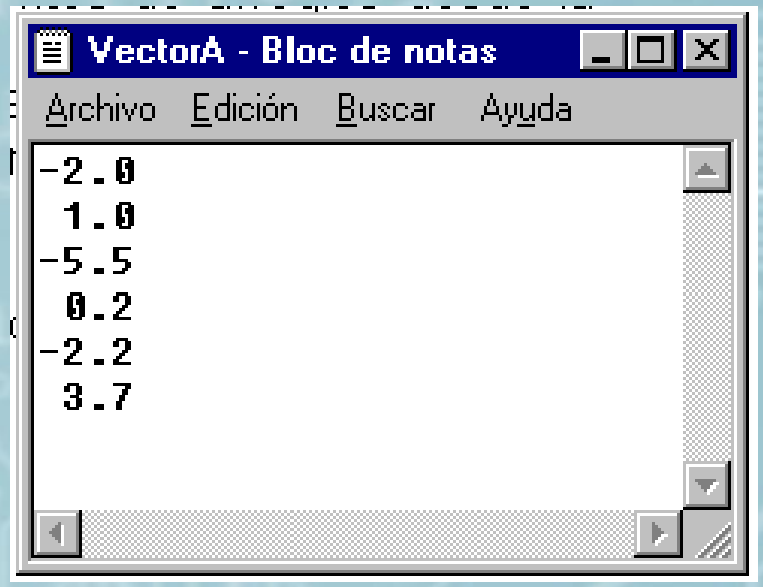
FORTRAN®

- **Arreglos.**
- **Lectura de archivos externos.**
- Este tipo de lectura tiene una gran importancia pues algunos softwares comerciales la utilizan para recibir la información que necesitan para realizar las tareas para las que fueron diseñados. Para esto se utilizan archivos de texto en los que se debe escribir la información siguiendo el formato requerido por el programa, se acostumbra escribir estos archivos de texto haciendo uso del programa Bloc de notas® incluido en el sistema operativo WINDOWS®.
- Es importante entender que la lectura de archivos externos no es exclusiva de los arreglos vectoriales o matriciales, se puede leer el valor de cualquier variable desde ellos, sin embargo es para los arreglos en donde se encuentra su mayor beneficio.

FORTRAN®

- **Arreglos.**
- **Lectura de archivos externos.**
- En la sección anterior se trató la lectura de los elementos de arreglos desde la pantalla de la computadora, para realizar esta lectura basta con que el usuario introduzca el valor y presione la tecla enter tantas veces como sea necesario, en este caso se recomienda que el archivo externo tenga las siguientes características:

Si se trata de la lectura de los elementos de un vector estos deberán escribirse en el archivo en forma de columna.



FORTRAN®

- **Arreglos.**
- **Lectura de archivos externos.**

  Si se trata de la lectura de los elementos de una matriz estos deberán estar escritos en forma de renglón – columna.

- Antes de tratar de leer del archivo externo es necesario abrirlo mediante la siguiente línea de código:

```
OPEN(UNIT=10,FILE="C:\VectorB.txt",STATUS="UNKNOWN")
OPEN(UNIT=11,FILE="C:\MatrizA.txt",STATUS="UNKNOWN")
```

FORTRAN®

- **Arreglos.**
- **Lectura de archivos externos.**

OPEN(UNIT=10,FILE="C:\VectorB.txt",STATUS="UNKNOWN")

OPEN(UNIT=11,FILE="C:\MatrizA.txt",STATUS="UNKNOWN")

- Es necesario asignar un número al archivo para identificarlo, este número es el que se indica con *UNIT=11*, se puede utilizar cualquier número mayor o igual a 10 para numerar al archivo, la ubicación del archivo dentro de la computadora o algún dispositivo externo (Disquete, CD, Memoria USB, etc.) se indica con *FILE="C:\MatrizA.txt"*, el programa se dirigirá a esta ruta para localizar el archivo, en caso de no encontrar el archivo o el dispositivo en donde se supone se encuentra, el programa no funcionara enviando un error, esto solo ocurre en la lectura, la extensión *.txt* del archivo no es obligatoria, se puede utilizar *.dat*, *.doc*, etc., la sentencia *STATUS="UNKNOWN"* se revisará en el tema de escritura de archivos externos.

FORTRAN®

- **Arreglos.**
- **Lectura de archivos externos.**

OPEN(UNIT=11,FILE="C:\MatrizA.txt",STATUS="UNKNOWN")

- Igual que con la memoria dinámica, en que es necesario desalojar las variables para poder redimensionar los arreglos con nuevos valores, es necesario cerrar los archivos en caso de requeris volver a acceder a ellos, esto se hace mediante la siguiente sentencia:

CLOSE(11)

- Así, para leer del archivo los valores del vector B y la matriz A la sintaxis son las siguientes:

```
DO I=1,N                    DO I=1,N
    READ(10,*)B(I)              READ(11,*)A(I,1:M)
END DO                      END DO
```

- El número del archivo se coloca en lugar del primer asterisco de los dos que acompañan la instrucción *READ(*,*)*, el funcionamiento en el caso de la matriz es exactamente el mismo, solo cambia la manera de escribirlo, si se desea, se puede pensar como un caso especial de la sentencia *DO*.

FORTRAN®

- **Arreglos.**
- **Impresión en archivos externos.**
- Para imprimir en archivos es necesario, igual que en la lectura, abrir el archivo deseado mediante la línea:

  OPEN(UNIT=10,FILE="C:\Resultados.txt",STATUS="UNKNOWN")

- Aquí la sentencia STATUS="UNKNOWN" es más importante que en la lectura, pues en caso de no existir el archivo en la ruta especificada el programa lo creará automáticamente. Las líneas para imprimir los datos contenidos en un arreglo dentro de un archivo externo son las mismas que en la de impresión en pantalla, solo se necesita agregar él número del archivo en la parte reservada para ello.

```
DO I=1,N
   WRITE(10,*)B(I)
END DO
```

```
DO I=1,N
      WRITE(11,*)A(I,1:M)
END DO
```

FORTRAN®

- **Arreglos.**
- **Ejercicio.** Escribir un programa que lea de un archivo externo la información de una matriz, incluyendo la dimensión, le sume un escalar y escriba la matriz resultado en otro archivo.

```
PROGRAM MATRIZ
IMPLICIT NONE
INTEGER::I,J
        !Contadores
INTEGER::N
        !Dimensión arreglos
REAL(8),ALLOCATABLE,DIMENSION(:,:)::A,B
        !Arreglos
OPEN(UNIT=10,FILE="C:\Matrixa.txt",STATUS="UNKNOWN")
OPEN(UNIT=11,FILE="C:\MatrixB.txt",STATUS="UNKNOWN")
OPEN(UNIT=12,FILE="C:\Matrizc.txt",STATUS="UNKNOWN")
READ(10,*)N
ALLOCATE(A(N,N),B(N,N))
DO I=1,N
READ(10,*)A(I,1:N)
END DO
DO I=1,N
READ(11,*)B(I,1:N)
END DO
…
…
DO I=1,N
WRITE(12,*)C(I,1:N)
END DO
DEALLOCATE(A,B,C)
CLOSE(10)
CLOSE(11)
CLOSE(12)
END PROGRAM
```

- En este ejemplo la lectura de la dimensión N de los arreglos se hace desde el archivo Matriza.txt, la lectura se realiza línea por línea, entonces una vez que el programa lee el valor de N cambia de línea, es en esta que se encuentra el primer renglón de la matriz A dentro del archivo.

FORTRAN®

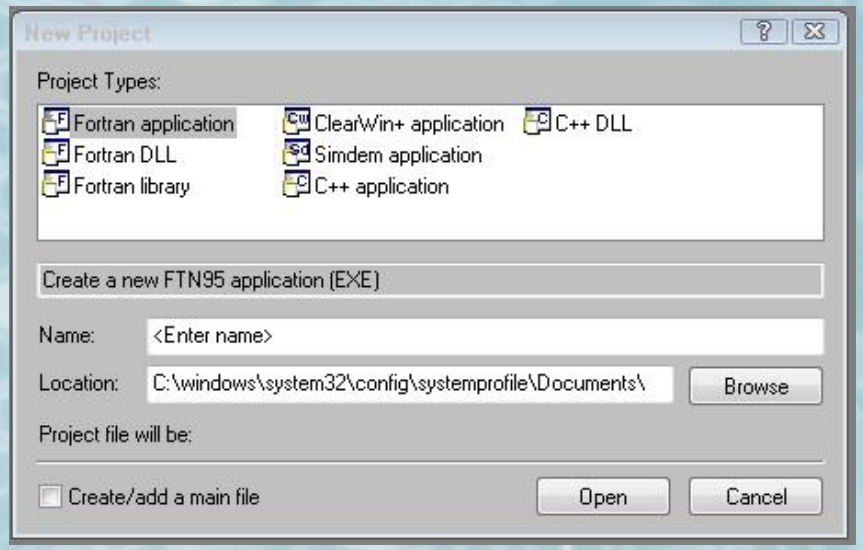
- **Programa #4: Matrices Parte 3.**
- Completar el programa número cuatro agregando a las siguientes opciones del menú la posibilidad de leer la información de las matrices desde un archivo de entrada e imprimir el resultado en un archivo de salida:
    6. Suma de dos matrices.
        I. Introducir e imprimir información en pantalla.
        II. Recibir e imprimir información en archivos externos.
    7. Producto de dos matrices.
        I. Introducir e imprimir información en pantalla.
        II. Recibir e imprimir información en archivos externos.
- Enviar el código, algoritmo y prueba de escritorio en un archivo de Word al correo de la materia a más tardar a las 19:00 horas del día anterior a la siguiente clase.
- El nombre del archivo debe seguir la siguiente estructura:
    - ***matrices3_primerapellido_segundoapellido_nombre***

FORTRAN®

- **Programación estructurada.**
- En ocasiones durante un programa es necesario realizar un mismo cálculo varias veces, o puede ocurrir que un programa necesite de otro previamente elaborado, para esto existen los conceptos de funciones, módulos y subrutinas.
- Un ejemplo sencillo son las permutaciones y combinaciones, ambas recurren al cálculo del factorial de un número, por lo que el programa que permita calcular el factorial se puede incluir dentro de una subrutina o un módulo.
- Para efectos de este curso trabajaremos exclusivamente con funciones y mayormente con subrutinas.

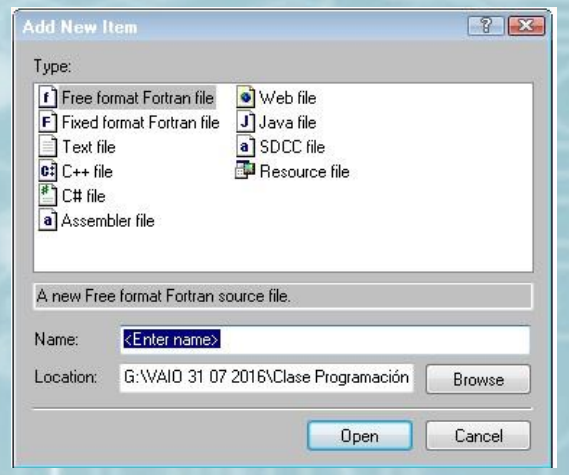
FORTRAN®

- **Programación estructurada.**
- Para poder programar las subrutinas que queremos utilizar en un programa principal debemos seguir algunos pasos. Cuando entramos al lenguaje, basta con ir al menú “FILE”, elegir la opción “NEW PROJECT”, y después seleccionar “Fortran application”, indicar el nombre del proyecto y la ruta en la que se va a guardar.



FORTRAN®

- **Programación estructurada.**
- Del lado derecho aparecerá una ventana llamada “Project Explorer” con tres carpetas, “Source Files”, “Include Files”, “Resource Files”. En la carpeta “Source Files” dar click derecho y seleccionar “Add New Item”, y en la ventana emergente seleccionar “Free format Fortran File” y nombrar el archivo. Con esto se tiene una nueva “hoja” para escribir el código de las subrutinas que se llamarán desde el principal.



FORTRAN®

- **Programación estructurada.**
- Una función devuelve un solo valor para una variable dada, este depende de las variables que se relacionan con la función, puede o no estar dentro de un módulo. La sintaxis en el programa principal es la siguiente:

```
PROGRAM PRINCIPAL
USE nombredelafunción
IMPLICIT NONE
…
WRITE(*,*)Pb(Rp,T,DENR_O,DENR_G)
…
END PROGRAM
```

- La línea *WRITE(*,*)Pb(Rp,T,DENR_O,DENR_G)* indica que se escriba el valor de la variable Pb que depende de las variables Rp, T, DENR_O y DENR_G, entendiendo que los valores de estas últimas debieron leerse o calcularse previamente.

FORTRAN®

- **Programación estructurada.**

```
FUNCTION Pb(Rp,T,DENR_O,DENR_G)
IMPLICIT NONE
REAL(KIND=2),INTENT(IN)::Rp,T,DENR_O,DENR_G
REAL(KIND=2)::Pb
Pb=18.2*(((Rp/DENR_G)**0.83)*(10**(0.00091*T-0.0125*(141.5/DENR_O-131.5)))-1.4)
RETURN
END FUNCTION
```

- Dentro del código del programa principal solo se declaran las variables a utilizar en la función (código de color rojo), en la línea ***REAL(KIND=2),INTENT(IN)::Rp,T,DENR_O,DENR_G*** la palabra ***INTENT(IN)*** indica si la o las variables son de entrada, en caso de ser de salida se utiliza la palabra ***INTENT(OUT)*** y si es de ambos tipos ***INTENT(INOUT)***, después, fuera del programa principal en una subrutina o en un módulo se repite lo anterior (código de color rojo) y se define la operación que evalúa la función (código de color negro).

FORTRAN®

- **Programación estructurada.**
- ***Subrutinas.***
- Las subrutinas a diferencia de las funciones pueden ser programas completos en donde se calculen una o más variables. El procedimiento para tener una subrutina dentro de un programa o módulo es exactamente el mismo que para una función, solo que se sustituye la palabra *FUNCTION* por *SUBROUTINE* la diferencia esta en el programa principal, pues aquí es necesario mandar llamar a la subrutina indicando las variables con las que se va a trabajar.

```
PROGRAM PRINCIPAL
USE nombre
IMPLICIT NONE
…
CALL SUBROUTINE FACTORIAL(n,FACT)
…
END PROGRAM
```

FORTRAN®

- **Programación estructurada.**
- ***Subrutinas.***

```
SUBROUTINE FACTORIAL(n,FACT)
IMPLICIT NONE
INTEGER::i,n,FACT
WRITE(*,*)"INDICA EL NUMERO DEL CUAL DESEAS EL FACTORIAL"
READ(*,*)n
FACT=1
DO i=1,n
FACT=FACT*i
END DO
WRITE(*,*)"EL FACTROIAL DE ",n," ES ",FACT
END SUBROUTINE FACTORIAL
```

FORTRAN®

- Para aprender a programar es necesario practicar durante horas con diferentes problemas, esto permitirá conocer la lógica de programación, cuando esto se logra, se es capaz de programar en casi cualquier lenguaje, solo es necesario conocer la sintaxis específica de cada uno de ellos y aplicar los conocimientos adquiridos.
- Este pequeño manual no es suficiente para desarrollar dichas habilidades, solo pretende ser una guía para quien comienza a programar.

GRACIAS

Ing. Juan Carlos Sabido Alcántara

Ingeniero Petrolero

Facultad de Ingeniería UNAM