Algunas funciones intrínsecas útiles para trabajar con números en Fortran 90

Seminario de Computación – Verano 2009

ABS(A)

Función real que devuelve un número (REAL o INTEGER) del mismo tipo que A. Nos devuelve el valor absoluto de A.

Funciones trigonométricas

COS(A) , SIN(A) , TAN(A) toman argumentos reales o complejos, y devuelven un resultado del mismo tipo que el argumento. A debe ser medido en radianes para que el resultado sea el esperado.

También contamos con las funciones trigonométricas inversas:

ACOS(A) , ASIN(A) , ATAN(A) y ATAN2(A,B) . ATAN es la función inversa de la tangente para A en el intervalo –π/2 , π/2. Por otro lado ATAN2(A,B) es la inversa de la tangente de A/B en el intervalo – π , π

Funciones logarítmicas y exponenciales.

EXP(A) , LOG(A) , LOG10(A) permiten calcular la función exponencial, el logaritmo en base e y el logaritmo en base 10 respectivamente.

SINH(A) , SINH(A) , TANH(A) En todos estos casos el argumento de la función puede ser real o complejo, mientras que el resultado es real.

AINT(A,KIND) Trunca el valor de **A** al entero más cercano, pero el resultado de la función sigue siendo un número real. **AINT(3.7)** es **3.0**, **AINT(-3.7)** es **-3.0**.

ANINT(A,KIND) Redondea el valor de **A** al entero más cercano, pero el resultado sigue siendo REAL. **AINT(3.7)** es **4.0** y **AINT(-3.7)** es **-4.0**

En este caso, **KIND** es una variable **INTEGER** que indica la precisión con la que se entregará el resultado (por ejemplo 4 u 8). Este argumento es opcional.

Funciones para pasar de variables REAL a INTEGER.

INT(A,KIND) Similar a **AINT**, pero el resultado final es **INTEGER.**

NINT(A,KIND) Esta función toma un argumento **REAL** y devuelve el **INTEGER** más próximo.

CEILING(A,KIND) Esta función toma un argumento **REAL** y devuelve el **INTEGER** más próximo.

FLOOR(A,KIND) Devuelve el entero más grande menor que **A**.

Variables para pasar de INTEGER a REAL.

REAL(A,KIND) En este caso el argumento de la función es un **INTEGER**, pero el resultado final es una variable **REAL** (el valor numérico del resultado es el mismo). Y el argumento opcional **KIND** es un **INTEGER** que determina la precisión del número real resultante.

Una alternativa es usar la función **DBLE(A)** que siempre entregará como resultado una variable **REAL** de precisión doble.

Otras funciones:

MAX(A1,A2,….,AN) Función que devuelve un resultado del mismo tipo que los argumentos (deben ser todos del mismo tipo). De todos los argumentos ingresados devuelve el máximo.

Analogamente tenemos **MIN(A1,A2,….,AN)**

MOD(A,P) Esta función calcula **A- P*INT(A/P)** Si **A** es divisible por **P**, entonces el resultado de esta función es 0. El resultado de la función es del mismo tipo que los argumentos ingresados (ambos deben ser del mismo tipo).

MOD(5,3) es **2**, **MOD(-5,-3)** es **-2**

SIGN(A,B) Los argumentos pueden ser **REAL** o **INTEGER** (deben ser del mismo tipo), y el resultado es del mismo tipo que los argumentos. El resultado consiste en el valor de **A** con el signo de **B**.

Aparte de las funciones para manipular números en **Fortran**, existen además numerosas funciones para manipular matrices, variables **CHARACTER** (que discutiremos en las próximas clases). Y otras más que no vamos a discutir en este curso.

En la clase anterior vimos funciones intrínsecas que nos dan información sobre la precisión con la que estamos trabajando.

Existen dos funciones que nos indican con que tipo de variables **REAL** e **INTEGER** deberíamos trabajar dada la precisión que necesitamos para resolver un problema dado.

SELECTED_INT_KIND(R) R sería el máximo exponente que necesitamos para resolver un problema (debe ser **INTEGER**) y el resultado de la función es un **INTEGER** que representa la precisión mínima que requerimos para lograr representar números como los que necesitamos en la máquina donde estamos trabajando.

SELECTED_REAL_KIND(P,R) Este caso es similar, pero para variables **REAL**, **P** es el número de dígitos significativos que necesitamos y **R** es el máximo exponente que necesitamos. La función nos devolverá un **INTEGER** que representa la mínima precisión requerida en las variables **REAL** para poder lograr nuestro objetivo.

```
PROGRAM ejemplo
IMPLICIT NONE
INTEGER , PARAMETER :: REAL_KIND=SELECTED_REAL_KIND(7,30)
REAL (KIND=REAL_KIND) :: X
….. !Garantizo que la precision es al menos la requerida independientemente de la maquina.
```

Funciones para utilizar con variables CHARACTER.

Concatenación de variables CHARACTER.

Las variables CHARACTER pueden ser concatenadas (unidas) mediante el operador "//"

Ejemplo

```
PROGRAM ejemplo
IMPLICIT NONE
CHARACTER(len=10)  :: nombre , apellido
CHARACTER(len=20)  :: nombre_completo
nombre="Juan"
apellido="Perez"
nombre_completo=nombre//apellido
WRITE(*,*)nombre_completo
STOP
END PROGRAM ejemplo
```

El ejemplo anterior va a dejar muchos espacios entre el nombre y el apellido, porque “Juan” y “Perez” tienen menos de 10 letras.

Entonces podemos reemplazar la asignación de nombre_completo por esta otra:

nombre_completo=nombre(1:4)//apellido(1:5)

Pero en este caso no queda ningún espacio entre medio… para solucionar esto podríamos insertar un espacio entre ambos.

nombre_completo=nombre(1:4)//” “//apellido(1:5) !El espacio es un character como !cualquier otro.

Pero este caso, solo funciona para nombres de 4 letras y apellidos de 5 letras, probemos este código con nombre=“Marcelo” y apellido=“Longobardi” .

Para que nuestro programa funcione con cualquier nombre necesitamos conocer de alguna manera el largo de la variable CHARACTER.

Función LEN:

LEN(A) donde A es una variable CHARACTER me da el largo de la variable. El problema es que incluye todos los espacios en blanco.

Calculemos en nuestro ejemplo LEN(nombre) o LEN(apellido) ¿Cuál es el resultado?

Existe una función que calcula el largo de una variable CHARACTER pero sin tener en cuenta los blancos que puedan existir al final de la variable.

LEN_TRIM(A) me da el largo de A sin tener en cuenta los blancos que existen a la derecha del último carácter “no espacio” de A.

Calculemos en nuestro caso LEN_TRIM(nombre)

Para aplicar esto en nuestro ejemplo podríamos hacer los siguiente:

```
PROGRAM ejemplo
IMPLICIT NONE
CHARACTER(len=10)  :: nombre , apellido
CHARACTER(len=20)  :: nombre_completo
INTEGER                  ::  largo_nombre , largo_apellido
nombre=“Juan”
apellido=“Perez”
largo_nombre=LEN_TRIM(nombre)
largo_apellido=LEN_TRIM(apellido)
nombre_completo=nombre(1:largo_nombre)//” “//apellido(1:largo_apellido)
WRITE(*,*)nombre_completo
STOP
END PROGRAM ejemplo
```

Que pasaría si en lugar de ingresar los nombres de la manera que los ingresamos, dejáramos espacios en blanco al principio del apellido:

nombre="Juan"

apellido=" Perez"

En este caso nuestro programa nuevamente dejaría demasiados blancos entre el nombre y el apellido. Para acomodar esta situación tenemos algunas funciones útiles:

ADJUSTL(A) Elimina los espacios en blanco al comienzo de la variable y los coloca al final de la misma.

En nuestro ejemplo para evitar que los blancos al comienzo de la variable nos traigan problemas podemos hacer los siguiente.

A continuación de la asignación del nombre y apellido modificamos las variables de la siguiente manera:

nombre=ADJUSTL(nombre)

apellido=ADJUSTL(apellido)

Otro ejemplo: Quiero recuperar el teléfono de estas dos líneas de información.

PROGRAM ejemplo
CHARACTER(len=100) :: info1, info2
info1="Nombre: Juan Perez , Tel: 4701-3432 , Edad: 44"
Info2="Nombre: Marcelo Longobardi , Tel: 4533-3445 , Edad: 50"

La dificultad está en que el teléfono no comienza siempre en el mismo carácter. El lugar donde está contenida la información depende del largo del nombre.

Lo que si sabemos es que a continuación de la palabra "Tel: " carácter de por medio comienza el número de teléfono que tiene una longitud de 9 caracteres.

Podemos usar la función SCAN para detectar donde se encuentra la palabra "Tel:" de info1 e info2.

INDEX(A,SET,BACK) A es la variable **CHARACTER** donde vamos a "buscar" la ocurrencia de la secuencia de caracteres **SET**. Si no encontramos dicha secuencia de caracteres entonces el resultado de la función es 0. De lo contrario el resultado nos indica el número de character en donde comienza la cadena de caracteres SET.

BACK es un argumento opcional. Puede ser **TRUE** o **FALSE**.

Retomamos nuestro ejemplo…

```
PROGRAM ejemplo
CHARACTER(len=100)  :: info1, info2
INTEGER                    :: comienzo_numero
CHARACTER(len=9)       :: telefono1 , telefono2
info1="Nombre: Juan Perez , Tel: 4701-3432 , Edad: 44"
Info2="Nombre: Marcelo Longobardi , Tel: 4533-3445 , Edad: 50"

!Busco donde comienza la cadena de caracteres ":"
comienzo_numero=INDEX(info1,"Tel:")
!Todavia no tengo lo que quiero, tengo donde comienza "Tel:"
comienzo_numero=comienzo_numero+5
WRITE(*,*)"El telefono 1 comienza en el caracter",comienzo_numero
!Guardo el telefono 1 en la variable correspondiente.

telefono1=info1(comienzo_numero:comienzo_numero+9)

WRITE(*,*)"El telefono 1 es: ",telefono1

!Procedemos de la misma manera para el telefono 2.
comienzo_numero=INDEX(info2,"Tel:")
!Todavia no tengo lo que quiero, tengo donde comienza "Tel:"
comienzo_numero=comienzo_numero+5
WRITE(*,*)"El telefono 2 comienza en el caracter",comienzo_numero
!Guardo el telefono 2 en la variable correspondiente.

telefono2=info2(comienzo_numero:comienzo_numero+9)

WRITE(*,*)"El telefono 2 es: ",telefono2


STOP
END PROGRAM ejemplo
```

Ahora consideremos que pasaría en el siguiente caso:
info1="Nombre: Juan Perez , Tel: 0-800-333-4000 , Edad: 44"

Supongamos que Juan Perez debido a su enorme popularidad puso un 0-800 con lo cual el largo del teléfono no es el que esperábamos.

```
PROGRAM ejemplo
IMPLICIT NONE
CHARACTER(len=100)     :: info1
INTEGER                :: comienzo_numero , final_numero
CHARACTER(len=20)       :: telefono1

info1="Nombre: Juan Perez , Tel: 0-800-333-4000 , Edad: 44"

comienzo_numero=INDEX(info1,"Tel:") + 5
final_numero=INDEX(info1,"Edad")-3

telefono1=info1(comienzo_numero:final_numero)

WRITE(*,*)"El telefono 1 es: ",telefono1


STOP
END PROGRAM ejemplo
```