6 CARACTERES Y CADENAS

6.1 Caracteres y cadenas

- En el capítulo 1 se indicó cómo tratar con variables y constantes carácter en Fortran 90/95. Recordar que la declaración de este tipo de variables, como ocurre con cualquier otro, puede incluir una inicialización de las mismas. Por otro lado, las constantes carácter pueden tener nombres simbólicos si se añade el atributo PARAMETER en su sentencia de declaración y deben encerrarse entre comillas dobles o simples.
- Posteriormente, en el capítulo 4 se estudió que la forma de declarar un array es idéntica para cualquier tipo de datos.
- Ejemplo. Escribir sentencias de declaración de diferentes datos carácter.

CHARACTER (len=15):: apellido ! declara variable carácter

CHARACTER:: seguir='S' ! declara e inicializa var carácter

CHARACTER (len=10), PARAMETER:: archivo='entrada' ! declara nombre simbólico para constante carácter

CHARACTER (len=25), DIMENSION(50)::alumnos ! declara array de 50 ! elementos cada uno de los cuales puede ser un conjunto de 25 !caracteres como máximo

- Una cadena de caracteres o simplemente *cadena* es una sucesión explícita de caracteres.
- Una subcadena de caracteres o simplemente *subcadena* es una porción contigua de caracteres de una cadena. Para referenciar una subcadena de una cadena, la sintaxis general es.

nombre ([pos_inicial]: [pos_final])

- *nombre* es el nombre de una variable o elemento de array carácter.
- *pos_inicial* es una expresión numérica que especifica la posición inicial (más a la izquierda) del primer carácter de la subcadena. Si no se especifica, se toma como valor por defecto la posición del primer carácter.
- *pos_final* es una expresión numérica que especifica la posición final (más a la derecha) del último carácter de la subcadena. Si no se especifica, se toma como valor por defecto la longitud de *nombre*.
- Los valores de *pos_inicial* y *pos_final* deben cumplir la condición:

$1 \leq pos\_inicial \leq pos\_final \leq$ longitud_nombre

- Ejemplo. Sea la declaración:

CHARACTER (len=10):: nombre='Susana'

nombre(2:4) hace referencia a ‘usa’

nombre(:) hace referencia a ‘Susana____’[7]

6.2 Expresión carácter

- Los operadores disponibles en Fortran 90/95 para operar con cadenas son:
    - el operador de concatenación // para concatenar cadenas. La sintaxis general de una expresión carácter que emplea este operador para concatenar variables carácter es:

var1_caracter // var2_caracter

    - Ejemplo. Sea la declaración:

CHARACTER (len=10):: c1,c2

c1 = ‘aero’

c2 = ‘plano’

WRITE(*,*) c1//c2 !se escribe por pantalla *aeroplano_*

    - los operadores relacionales (ver Tabla 2.1) para comparar cadenas. Sin embargo, su uso está desaconsejado pues el resultado de la comparación puede variar de computador a computador. En su lugar, se aconseja comparar cadenas utilizando las funciones intrínsecas léxicas cuyos resultados son independientes del procesador. Estas funciones se estudian en la sección 6.4.
    - Ejemplo. Sean las declaraciones:

CHARACTER (len=15):: apellido1,apellido2

apellido1 < apellido2 !Expresión carácter que compara las 2 variables

6.3 Asignación carácter

- Una sentencia de asignación carácter asigna el valor de una expresión carácter a una variable o elemento de array del mismo tipo. La sintaxis general es:

variable_carácter = expresión_carácter

- El funcionamiento es:
    - Se evalúa la expresión carácter.

[7] Cada guión bajo representa un blanco.

  - Se asigna el valor obtenido a la variable carácter.
  - Si la longitud de la variable es mayor que la de la expresión, el valor de la expresión se ajusta a la izquierda de la variable y se añaden blancos hasta completar la longitud total de la variable.
  - Si la longitud de la variable es menor que la de la expresión, el valor de la expresión es truncado.

- Ejemplo. Sea la declaración:

CHARACTER (len=10):: c1,c2,c3

c1 = ‘aero’

c2 = ‘plano’

c3=c1//c2

WRITE(*,*) c3 !se escribe por pantalla *aeroplano_*

c3=’demasiado largo’

WRITE(*,*) c3 !se escribe por pantalla *demasiado_*

6.4 Funciones intrínsecas carácter

- A continuación, se definen algunas funciones intrínsecas útiles para manipular caracteres: IACHAR, ACHAR, LEN, LEN_TRIM TRIM, INDEX.
  - IACHAR(caracter) convierte el carácter de entrada en un número que corresponde a su posición en el código ASCII.
  - ACHAR(numero) es la función inversa de IACHAR, pues convierte el número de entrada en un carácter según su posición en el código ASCII.
  - LEN(cadena) devuelve la longitud declarada para la variable carácter.
  - LEN_TRIM(cadena) devuelve un número entero que corresponde a la longitud de la cadena eliminando los blancos.
  - TRIM(cadena) devuelve la cadena eliminado los blancos.
  - INDEX(cadena1,subcadena2[,.TRUE.]) devuelve la primera coincidencia del patrón subcadena2 en la cadena1. Si el tercer argumento no está, la búsqueda se realiza de izquierda a derecha. Si el tercer argumento está presente, la búsqueda se realiza de derecha a izquierda. Si no se encuentra coincidencia devuelve un 0.

- Ejemplo.

CHARACTER (len=10)::pal1=’raton’

WRITE(*,*) IACHAR(‘A’),IACHAR(‘Z’),IACHAR(‘a’),IACHAR(‘z’)

!se escribe por pantalla 65,90,97,122

```
WRITE(*,*) ACHAR(65),ACHAR(90),ACHAR(97),ACHAR(122)
!se escribe por pantalla A,Z,a,z
WRITE(*,*) LEN(pal1),LEN_TRIM(pal1)
!se escribe por pantalla 10   5
WRITE(*,*) pal1,TRIM(pal1)
!se escribe por pantalla raton_____raton
WRITE(*,*) INDEX(pal1,'a')
!se escribe por pantalla 2
```

- Las funciones intrínsecas *léxicas* permiten comparar cadenas y son las siguientes: LLT (Lexically Less Than), LLE (Lexically Less or Equal than), LGT (Lexically Great Than) y LGE (Lexically Great or Equal than). Estas funciones son equivalentes a los operadores relaciones <, <=, > y >=, respectivamente. Ahora bien, mientras las funciones léxicas utilizan siempre el código ASCII como base para realizar las comparaciones, los operadores relacionales pueden utilizar este código o cualquier otro, según el computador.
- La comparación entre cadenas se realiza de la siguiente manera:
    - Se comparan las dos cadenas carácter a carácter, comenzando por el primer carácter (el que se encuentra más a la izquierda) y continuando hasta que se encuentra un carácter distinto o hasta que finaliza la cadena.
    - Si se encuentran caracteres distintos, el operando que contiene el carácter menor[8], será considerado el operando menor. Por tanto, el orden de los operandos viene marcado por el primer carácter que difiere entre ambos operandos.
    - Si se alcanza el final de uno de los operandos y no hay caracteres distintos, la ordenación de las cadenas se hará en función de sus longitudes. Así, si ambas cadenas tienen la misma longitud, las cadenas son iguales, mientras que si las cadenas tienen longitudes diferentes, la comparación continúa como si la cadena más corta estuviera rellena de blancos hasta la longitud de la cadena más larga.
- Ejemplo. Sean las declaraciones:

```
CHARACTER (len=15):: pal1,pal2
```

[8] Un carácter es menor que otro si la posición que ocupa el primero en el código ASCII es menor que la que ocupa el segundo.

```
LOGICAL:: result1,result2
pal1=’Begoña’
pal2=’Paula’
result1=pal1<pal2
result2=LLT(pal1,pal2)
```

El valor de result1 puede variar de procesador a procesador, pero el valor de result2 es siempre .TRUE. en cualquier procesador.

EJERCICIOS RESUELTOS

Objetivos:

Aprender a usar variables y arrays carácter en Fortran y a transferirlos a procedimientos externos. Manejar las funciones intrínsecas más importantes relacionadas con este tipo de variables.

1. Pedir el nombre y apellido de una persona. Hallar la longitud de la cadena nombre. Guardar el nombre completo en una única variable y decir cuantas ‘aes’ tiene.

```
PROGRAM cap6_1
IMPLICIT NONE
CHARACTER (LEN=10) :: nom, apel
CHARACTER (LEN=20) :: nomc
INTEGER :: long1,long2=0,i,conta=0

WRITE(*,*) 'DAME TU NOMBRE'
READ(*,*) nom

long1=len_trim(nom)
WRITE(*,*) 'nom,' tiene ',long1,’caracteres’
!otro modo de calcular la longitud de una cadena
DO i=1,LEN(nom)
  IF (nom(i:i) /= ' ') THEN
    long2=long2+1
  END IF
END DO
WRITE(*,*)'nom,' tiene ',long2,’caracteres’
!.....
WRITE(*,*) 'DAME TU APELLIDO'
READ(*,*) apel

nomc=TRIM(nom)//' '//apel
WRITE(*,*) 'TU NOMBRE COMPLETO ES ',nomc

DO i=1,LEN_TRIM(nomc)
  IF (nomc(i:i)=='A'.OR. nomc(i:i) == 'a') THEN
    conta=conta+1
  END IF
END DO
WRITE(*,*) 'LA CANTIDAD DE A EN ',nomc,' ES',conta

END PROGRAM cap6_1
```

– El operador de concatenación // permite concatenar el nombre y apellido en una expresión carácter y asignar el resultado a la variable nomc.

– Repasar en la sección 6.1 la forma de referenciar una subcadena de una cadena. En este ejercicio, para extraer de una en una las letras de la variable nomc, se escribe nomc(i:i), con la posición inicial igual a la posición final, dentro de un bucle, en el que el índice i toma los valores desde 1 hasta la longitud de la cadena nombre.

2. Escribir por pantalla la tabla de caracteres ASCII usando la función intrínseca IACHAR.

```
PROGRAM cap6_2
IMPLICIT NONE
CHARACTER (LEN=27) :: abc='ABCDEFGHIJKLMNÑOPQRSTUVWXYZ',&
abcm='abcdefghijklmnñopqrstuvwxyz'
INTEGER:: i
WRITE(*,*) 'N ASCII    LETRA   N ASCII    LETRA'
DO i=1,27
  WRITE(*,100) IACHAR(abc(i:i)),abc(i:i),IACHAR(abcm(i:i)),abcm(i:i)
PAUSE
END DO
100 FORMAT(2X,I4,8X,A2,8X,I4,6X,A2)
END PROGRAM cap6_2
```

– La sentencia PAUSE suspende temporalmente la ejecución del programa.

– La sentencia FORMAT permite mostrar con formatos específicos la lista de variables dada de forma que éstas quedan en columnas bien alineadas. Esta sentencia se explica en detalle en el capítulo 7.

– ¿Qué números corresponden a las letras ñ y Ñ?

– ¿Qué relación existe entre el número asociado a una letra minúscula y su correspondiente mayúscula?

3. Pasar a minúsculas un nombre que se lee por teclado usando las funciones intrínsecas IACHAR y ACHAR. Suponer que el nombre leído puede tener mezcladas letras minúsculas y mayúsculas.

```
PROGRAM cap6_3
IMPLICIT NONE
```

```
CHARACTER (LEN=20) :: nom=' ',nom_mayus
INTEGER :: num,num_mayus,i
WRITE(*,*) 'DAME UN NOMBRE EN MAYUSCULAS'
READ(*,*) nom_mayus
WRITE(*,*) 'EL NOMBRE TECLEADO ES ',nom_mayus
DO i=1,LEN_TRIM(nom_mayus)
   num_mayus=IACHAR(nom_mayus(i:i))
   IF (num_mayus >= 65.AND.num_mayus <= 90) THEN
     num=num_mayus+32
     nom(i:i)=ACHAR(num)
   ELSE
     nom(i:i)=ACHAR(num_mayus)
   END IF
END DO
WRITE(*,*) 'EL NOMBRE EN MINUSCULAS ES ',nom
END PROGRAM cap6_3
```

– Para saber si cada letra del nombre es mayúscula, se obtiene su número asociado y se testea si pertenece al intervalo [65-90] (se prescinde de la ñ). En caso afirmativo, se suma 32 al número, se reconvierte a letra y se coloca en la posición adecuada de la variable declarada para almacenar el nombre en minúsculas.

4. Pasar a mayúsculas un nombre que se lee por teclado usando las funciones intrínsecas IACHAR y ACHAR. Suponer que el nombre leído puede tener mezcladas letras minúsculas y mayúsculas.

```
PROGRAM cap6_4
IMPLICIT NONE
CHARACTER (LEN=20) :: nom,nom_mayus=' '
INTEGER :: num,num_mayus,i
WRITE(*,*) 'DAME UN NOMBRE EN MINUSCULAS'
READ(*,*) nom
WRITE(*,*) 'EL NOMBRE TECLEADO ES ',nom
DO i=1,LEN_TRIM(nom)
 num=IACHAR(nom(i:i))
 IF (num >= 97 .AND. num <= 122) THEN
    num_mayus=num-32
    nom_mayus(i:i)=ACHAR(num_mayus)
 ELSE
```

```
    nom_mayus(i:i)=ACHAR(num)
  END IF
END DO
WRITE(*,*) 'EL NOMBRE EN MAYUSCULAS ES ',nom_mayus
END PROGRAM cap6_4
```

– Se usa el mismo procedimiento que en el ejercicio anterior. Para pasar a mayúsculas una letra minúscula basta restar 32 al número correspondiente según la tabla ASCII.

– ¿Qué ocurre si restamos un valor constante distinto de 32 a cada número? ¿puede servir este método para encriptar mensajes?

5. Invertir una palabra usando una subrutina. La palabra invertida debe almacenarse en la misma variable usada para la palabra original.

```
PROGRAM cap6_5
IMPLICIT NONE
CHARACTER (LEN=50) :: nombre
INTEGER :: long,i=0
WRITE(*,*) 'DAME UN NOMBRE'
READ(*,*) nombre

DO
  i=i+1
  IF (nombre(i:i) == ' ') EXIT
END DO
WRITE(*,*) 'LA PALABRA TIENE',i-1,' CARACTERES'
long=i-1
CALL invertir(nombre,long)
WRITE(*,*) 'LA PALABRA INVERTIDA ES ',nombre
END PROGRAM cap6_5

SUBROUTINE invertir(nombre,long)
IMPLICIT NONE
INTEGER, INTENT(IN) :: long
CHARACTER (LEN=long), INTENT(IN OUT) :: nombre
CHARACTER (LEN=1) :: aux
INTEGER :: cen,i,j
```

```
j=long
cen=long/2
DO i=1,cen
  aux=nombre(i:i)
  nombre(i:i)=nombre(j:j)
  nombre(j:j)=aux
  j=j-1
  WRITE(*,*) 'NOMBRE ',nombre
END DO
END SUBROUTINE invertir
```

– El algoritmo usado para la inversión consiste en localizar la posición central de la palabra e intercambiar las posiciones de las letras última y primera, penúltima y segunda y así sucesivamente hasta llegar a la posición central de la palabra.

– El bucle DO del programa principal permite calcular la longitud de la palabra a invertir. La función intrínseca LEN_TRIM (cadena) realiza la misma tarea.

– Al llamar a la subrutina, se transfiere la dirección de memoria del primer carácter de la variable nombre.

6. Leer una sílaba y una palabra y escribir en qué posición de la palabra está la sílaba si es que está, empezando por la izquierda.

```
PROGRAM cap6_6
IMPLICIT NONE
CHARACTER (LEN=6) :: sil=' '
CHARACTER (LEN=30) :: pal=' '
WRITE(*,*) 'PALABRA'
READ(*,*) pal
WRITE(*,*) 'SILABA'
READ(*,*) sil
WRITE(*,*) INDEX(pal,TRIM(sil))
END PROGRAM cap6_6
```

7. Buscar un nombre en un array de cuatro nombres.

```
PROGRAM cap6_7
IMPLICIT NONE
```

```
CHARACTER (LEN=15), DIMENSION(4) :: &
nom=(/'PEPE GOTERA','ROMPETECHOS','MORTADELO ','FILEMON   '/)
CHARACTER (LEN=15):: busca
INTEGER :: switch,i=0,ipos

WRITE(*,*) 'DAME UN NOMBRE'
READ(*,*) busca
switch=0
DO
 i=i+1
 IF (nom(i) == busca) THEN
  switch=1
  ipos=i
  WRITE(*,*) 'EL NOMBRE SE ENCUENTRA EN LA POSICION',ipos
 END IF
 IF (i == 4 .OR. switch == 1) EXIT
END DO
IF (switch == 0) THEN
 WRITE(*,*) 'EL NOMBRE NO ESTA EN LA LISTA'
END IF
END PROGRAM cap6_7
```

– En este programa nom es el identificador de un array de 4 componentes, cada una de las cuales es un nombre de 15 caracteres como máximo. Todos los elementos deben tener la misma longitud.

– El algoritmo usado en este ejercicio es el mismo que el del programa cap4_3 para buscar un número en un vector de números. Al igual que allí, la variable switch funciona de interruptor, de modo que el programa actúa según su contenido.

8. Inicializar los códigos y los nombres de diez provincias en dos vectores carácter. A continuación, se pide al usuario un código por teclado y el programa debe mostrar el nombre de la provincia correspondiente. Si no existe el código leído, mostrar un mensaje que avise de ello. El programa se ejecuta mientras el usuario lo desee.

```
PROGRAM cap6_8
IMPLICIT NONE
INTEGER:: i
CHARACTER (LEN=1) :: resp
```

```
CHARACTER(LEN=2):: cod
CHARACTER (LEN=2), DIMENSION(10) :: &
tcod=(/'A ','AL','AV','B ','BA','C ','CA','CC','CO','CS'/)
CHARACTER (LEN=9),DIMENSION(10) :: &
tnom=(/'ALICANTE ','ALMERIA  ','AVILA    ','BARCELONA', 'BADAJOZ
'&
,'CORUÑA  ','CADIZ   ','CACERES ','CORDOBA ', 'CASTELLON'/)

DO
 WRITE(*,*) 'DAME UN CODIGO DE PROVINCIA'
 READ(*,*)cod
! ***BUSQUEDA EN EL ARRAY
 i=0
 DO
  i=i+1
      IF (cod == tcod(i) .OR. i == 10) EXIT
 END DO
 IF (cod /= tcod(i)) THEN
  WRITE(*,*) 'ERROR. NO EXISTE ESE CODIGO'
 ELSE
  WRITE(*,*) 'LA PROVINCIA ES ',tnom(i)
 END IF
 WRITE(*,*) 'CONTINUAR(S/N)?'
 READ(*,*) resp
 IF (resp /= 'S' .AND. resp /= 's') EXIT
END DO
END PROGRAM cap6_8
```

9. Ordenar ascendentemente los nombres de tres personas que se introducen por teclado. Utilizar el método de la burbuja.

```
PROGRAM cap6_9
IMPLICIT NONE
INTEGER, PARAMETER::N=3
CHARACTER (LEN=20), DIMENSION(N) :: nom
INTEGER:: i
CALL leer(nom,N)
CALL ordenar(nom,N)
WRITE(*,*) 'LA LISTA ORDENADA ASCENDENTEMENTE ES'
WRITE(*,*) (nom(i),i=1,N)
```

```
END PROGRAM cap6_9

SUBROUTINE leer(x,tam)
IMPLICIT NONE
INTEGER, INTENT(IN):: tam
CHARACTER (LEN=20), DIMENSION(tam), INTENT(OUT) :: x
INTEGER:: i
DO i=1,tam
  WRITE(*,*) 'DAME NOMBRE ENTRE APOSTROFES',i
  READ(*,*) x(i)
END DO
END SUBROUTINE leer

SUBROUTINE ordenar(x,n)
IMPLICIT NONE
INTEGER, INTENT(IN):: n
CHARACTER (LEN=*), DIMENSION(n), INTENT(IN OUT) :: x
CHARACTER (LEN=20)::aux
INTEGER:: i,j
  DO  i=n-1,1,-1
  DO  j=1,i
    IF (LGT(x(j),x(j+1))) THEN
      aux=x(j)
      x(j)=x(j+1)
      x(j+1)=aux
    END IF
  END DO
END DO
END SUBROUTINE ordenar
```

– El método de la burbuja ya se implementó para ordenar números en el capítulo 4 (ver cap4_7).

– ¿Cómo cambia el programa si se quiere realizar un ordenamiento descendente y el número de personas es siete?

EJERCICIOS PROPUESTOS

1) Programa que codifica la frase 'EXAMEN DE INFORMATICA'.

2) Programa que descodifica la frase codificada en el ejercicio anterior. Utiliza una subrutina para codificar y descodificar la frase.

3) Programa que lea dos palabras y las muestre ordenadas alfabéticamente. (No usar ningún método de ordenamiento).

4) Programa que pida una frase y cuente el número de palabras.

5) Programa que lea una palabra y verifique si es un palíndromo o no. Utilizar el programa cap6_5 para invertir la palabra. Usar una función lógica para determinar si la palabra dada es un palíndromo o no.

6) Programa que lea una frase y cuente el número de vocales de cada palabra mostrando la de mayor número por pantalla. Usa una subrutina para leer la frase y otra para determinar la palabra que contiene más vocales.

7) Programa que lea una palabra aguda y diga si requiere tilde o no según las reglas de acentuación. El programa se ejecuta hasta que el usuario introduzca la palabra FIN. Usa programación modular para la construcción del programa.

8) Programa que lea una palabra y el idioma en que está escrita (inglés/castellano) y muestre su traducción (castellano/inglés). Suponer que el diccionario está formado por las palabras siguientes: Computador, raton, pantalla, teclado, programa, ejecutar. Computer, mouse, screen, keyboard, program, execute. Usa programación modular para la construcción del programa.