Guía práctica de estudio 13: Lectura y escritura de datos

Elaborado por:
Ing. Jorge A. Solano Gálvez
Guadalupe Lizeth Parrales Romay

Revisado por:
M.C. Edgar E. García Cano

Autorizado por:
M.C. Alejandro Velázquez Mena

Guía práctica de estudio 13: Lectura y escritura de datos

Objetivo:

Elaborar programas en lenguaje FORTRAN que requieran el uso de archivos de texto plano en la resolución de problemas:

Actividades:

- Abrir y cerrar un archivo.
- Leer y escribir en un archivo.

Introducción

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Lenguaje FORTRAN permite manejar la entrada y la salida de datos desde o hacia un archivo, a través de funciones intrínsecas.

Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Jorge A. Solano
 *
*/
```

Número de unidad

En FORTRAN cada archivo está asociado a un número de unidad, que es un entero entre 1 y 99. Algunos números de unidad están reservados, por ejemplo: 5 para la entrada estándar (teclado) y 6 para la salida estándar (pantalla).

Abrir archivo

La función *open()* se utiliza para abrir archivos, esto permite que FORTRAN pueda leer o escribir en él. Su estructura es la siguiente:

open(lista_de_especificadores)

Los especificadores más comunes son:

Especificador	Descripción	
[UNIT=]	*u*	Número de unidad. Un número entero entre 1 y 99 (Se puede elegir cualquier número pero éste debe ser único).
IOSTAT=	*ios*	Identificador de estado de E/S. Debe ser una variable entera. Devuelve cero si la operación fue exitosa y cualquier otro número en caso de error.
ERR=	*err*	Error. Etiqueta a la que el programa saltará si ocurre un error.
FILE=	*fname*	Nombre del archivo. Cadena de caracteres que denota el nombre del archivo.
STATUS=	*sta*	Modo de Apertura. Cadena de caracteres que puede tener los siguientes valores: • NEW: Crear un archivo nuevo. • OLD: Abre un archivo existente. • SCRATCH: Crea un archivo que existe solo durante la ejecución del programa y que se destruye con la instrucción de cerrar archivo (o cuando el programa termina).
ACCESS=	*acc*	Tipo de Acceso. Cadena de caracteres que puede tener los siguientes valores: • SEQUENTIAL • DIRECT Por defecto, el tipo de acceso es SEQUENTIAL.
FORM=	*frm*	Formato. Puede tomar los siguientes valores: • FORMATTED • UNFORMATTED El formato por defecto es UNFORMATTED.
RECL=	*rl*	Especifica la longitud de cada registro en un archivo de acceso directo.

Cerrar archivo

La función *close()* permite cerrar uno o varios archivos que fueron abiertos mediante una llamada a open(). La función *close()* permite escribir la información que se encuentre en el buffer hacia el disco y realiza un cierre formal del archivo a nivel del sistema operativo.

Un error en el cierre de un archivo puede generar todo tipo de problemas, incluyendo pérdida de datos, destrucción de archivos y posibles errores intermitentes en el programa. La estructura de esta función es:

close(número_de_unidad[,lista_de_parametros])

Parámetro	Descripción	
[UNIT=]	*u*	Número de unidad, un entero único entre 1 y 99.
IOSTAT=	*ios*	Identificador de estado de E/S. Debe ser una variable entera. Devuelve cero si la operación fue exitosa y cualquier otro número en caso de error.
ERR=	*err*	Error. Etiqueta a la que el programa saltará si ocurre un error.
STATUS=	*sta*	Modo de Cerrado. Cadena de caracteres que puede tener los valores: • KEEP: Valor por defecto. • DELETE.

Los parámetros entre corchetes son opcionales. Si IOSTAT devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

Código (abrir y cerrar archivo existente)

```
      program abrirCerrarArchivoExistente

c Este programa permite abrir un archivo en modo lectura.

      integer error, u
      parameter(u=20)

      open(u, FILE='archivo.txt', STATUS='OLD', IOSTAT=error)

      if (error .EQ. 0) then
         write (*,*) 'El archivo se abrió correctamente'
         close(u)
      else
         write (*,*) 'Error al abrir el archivo', error
         write (*,*) 'El archivo no existe o no se'
         write (*,*) 'tienen permisos de lectura.'
      endif

      stop
      end program
```

El programa anterior intenta abrir un archivo que ya existe, esto se especifica en el parámetro STATUS='OLD'. Si se desea abrir un archivo que no existe, es decir, crear un nuevo archivo, se debe especificar en el parámetro STATUS='NEW'.

Código (abrir y cerrar archivo existente)

```
      program abrirCerrarArchivoNuevo

c Este programa permite abrir un archivo en modo lectura.

      integer error, u
      parameter(u=20)

      open(u, FILE='archivo.txt', STATUS='NEW', IOSTAT=error)

      if (error .EQ. 0) then
         write (*,*) 'El archivo se abrió correctamente'
         close(u)
      else
         write (*,*) 'Error al abrir el archivo', error
         write (*,*) 'El archivo ya existe o no se'
         write (*,*) 'tienen permisos de escritura.'
      endif

      stop
      end program
```

Funciones READ y WRITE

Las funciones READ y WRITE permiten leer y escribir, respectivamente, datos sobre los archivos. Las estructuras de estas funciones son, respectivamente:

read([UNIT=] u, [FMT=] fmt, [IOSTAT=ios, ERR=err, END=s])
write([UNIT=] u, [FMT=] fmt, [IOSTAT=ios, ERR=err, END=s])

Donde el especificador END=s define a que etiqueta debe saltar el programa si se alcanza el fin del archivo. El especificador FMT permite utilizar los descriptores de formato.

La función *write()* permite escribir cualquier tipo de dato en un archivo especifico. La función *read()* permite leer cualquier tipo de dato desde el archivo especificado. Esta función lee un renglón a la vez.

Código (read)

```
      program leerDeArchivo

c Este programa permite leer el contenido de un archivo.

      integer error, u
      character (len=5) caracteres
      parameter(u=11)

      open(u, FILE= 'archivo.txt', STATUS='OLD', IOSTAT=error)

      if (error .EQ. 0) then
         write (*,*) 'El archivo se abrió correctamente'
         write (*,*) 'El contenido del archivo es: '
         do while (error .EQ. 0)
            read (u,*,IOSTAT=error) caracteres
            write (*,*) caracteres
         enddo
         close(u)
      else
         write (*,*) 'Error al abrir el archivo', error
         write (*,*) 'El archivo no existe o no se'
         write (*,*) 'tienen permisos de lectura.'
      endif

      stop
      end program
```

Tanto el número de unidad como el descriptor de formato son parámetros obligatorios para la función READ.

El número de unidad le indica a la función READ de donde va a obtener la información, en este caso se obtendrá del archivo abierto. El descriptor de formato permite imprimir la información en un formato personalizado, en este caso no está especificado y se indica con *.

Debido a que la función READ lee una línea cada vez, la capacidad del arreglo donde se almacena la información (en este caso caracteres) debe ser del tamaño máximo de caracteres por reglón, en otro caso la información (el renglón) ser verá truncado.

Código (write)

```
      program escribirEnArchivo

c Este programa permite escribir datos en un archivo.

      integer error, u
      character (len=35) escribir
      parameter(u=15)
      escribir = 'Escribir ésta cadena en archivo.'

      open(u, FILE='escribirCadena.txt', STATUS='NEW', IOSTAT=error)

      if (error .EQ. 0) then
         write (*,*) 'El archivo se creó correctamente'
         write (u,*) escribir
         close (u)
      else
         write (*,*) 'Error al crear el archivo', error
         write (*,*) 'El archivo ya existe o no se'
         write (*,*) 'tienen permisos de escritura.'
      endif

      stop
      end program
```

Código (Descriptores de formato)

```
      program formato

c Este programa permite leer diferentes tipos de datos
c con un formato específico desde un archivo.

      integer enteroUno, error, u
      real realUno
      complejo complejoUno
      character caracterUno

      parameter(u=9)

      open(u, FILE='variables.txt', STATUS='OLD', IOSTAT=error)

      if (error .EQ. 0) then
         read (u,100) enteroUno, realUno, caracterUno
  100    format (I3,F6.3,A2)
         write (*,*) 'Entero: ', enteroUno
         write (*,*) 'Real: ', realUno
         write (*,*) 'Carácter: ', caracterUno
         close (u)
      endif

      stop
      end program
```

En el archivo de texto, las variables deben estar separadas por espacios para que puedan ser leídas de manera correcta por el programa, es decir:

Texto (variables.txt)

```
23 35.456 x
```

Bibliografía

- Oracle (2010). Fortran 77 Languaje Reference. Consulta: Julio de 2015. Disponible en: http://docs.oracle.com/cd/E19957-01/805-4939/

- Stanford University (1995). Fortran 77 Tutorial. Consulta: Julio de 2015. Disponible en: http://web.stanford.edu/class/me200c/tutorial_77/