Tema 10: Estructuras de control de entrada y salida

Estructuras de control de entrada y salida (input, disp, fopen, fclose, fscanf, fprintf, textread, load, save).

Prof. Saúl. Buitrago y Oswaldo Jiménez

Instrucciones de entrada (lectura) y de salida (escritura)

Se verá a continuación una forma sencilla de leer variables desde teclado y escribir mensajes en la pantalla del PC. Más adelante se considerarán otros modos más generales y complejos de hacerlo.

- Función input

  Esta permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario.

  >> x = input('texto')

  Entre comillas debe escribirse el texto que queremos que lea.

```
u = input('var ')
var   7
u =
     7
```

```
u = input('var ')
var   [ 1 2 3 ]
u =
     1     2     3
```

```
u = input('var ')
var   'esto'
u =
esto
```

```
>> x = input('texto', 's')   →   u = input('var ',’s’)
                                 var   esto es
                                 u =
                                 esto es
```

El texto tecleado se lee y se almacena como texto sin evaluar

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Función disp

  Esta permite imprimir en pantalla un mensaje de texto o el valor de una variable, pero sin imprimir su nombre

  >> disp(‘el programa ha terminado’)

  >> a = rand(4,4);

  >> disp(a);

```
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
```

Estructuras de control de entrada (lectura) y de salida (escritura)

A continuación veremos funciones para lectura y escritura de archivos.

- Funciones fopen y fclose

  Estas funciones sirven para abrir y cerrar archivos, respectivamente.

  La función fopen tiene la forma:

  [fi, texto] = fopen(nom_archivo, c)

  donde **fi** es una variable que recibe el valor de retorno que identifica al archivo de nombre nom_archivo, **texto** es un mensaje para el caso de que se produzca un error, y **c** es un carácter (o dos) que indica el tipo de operación que se desea realizar con el archivo. Las opciones más importantes para **c** son las siguientes:

  'r' lectura (de read)

  'w' escritura reemplazando (de write)

  'a' escritura a continuación (de append)

  'r+' lectura y escritura

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Funciones fopen y fclose (cont.)

Cuando por alguna razón el archivo no puede ser abierto, se devuelve un -1 en la variable fi. En este caso el valor de retorno texto puede proporcionar información sobre el tipo de error que se ha producido.

Después de realizar las operaciones de lectura y escritura deseadas, el archivo se puede cerrar con la función fclose en la forma siguiente:

```
st = fclose( fi )
```

donde st es un valor de retorno para posibles condiciones de error, (cuando st retorna como 0 significa que el archivo cerró bien).
Si se quieren cerrar a la vez todos los archivos abiertos puede utilizarse el comando:

```
st = fclose( 'all' )
```

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Funciones fscanf y fprintf

  Estas funciones permiten leer y escribir en archivos ascii, es decir, en archivos formateados.

  Forma general de la función fscanf:

  [var, count] = fscanf(fi, format, size)

  Lee del archivo identificado por **fi** de acuerdo a un formato especificado por **format** y lo devuelve en la variable **var** (real, vector o matriz).

  count (opcional) devuelve el número de elementos leídos satisfactoriamente.

  size (opcional) limita el número de elementos a ser leidos en el archivo Si no se coloca, lee hasta el final del archivo. Si se especifica, se tienen las siguientes opciones

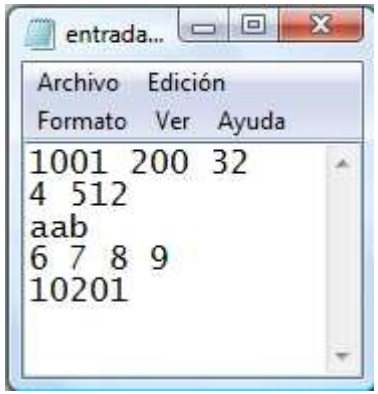
N	lee a lo más N elementos en un vector columna
inf	lee hasta el final del archivo
[M,N]	lee a lo más M * N elementos llenando al menos una matriz M×N siguiendo el orden de columnas. N puede ser inf, pero no M.

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Funciones fscanf y fprintf (cont.)

  format va encerrada entre comillas simples, y contiene los especificadores de formato para las variables

  %s para cadenas de caracteres
  %d para variables enteras
  %f para variables de punto flotante
  %lf para variables de doble precisión



Ejemplos:

```
>> fi1 = fopen('entrada1.txt', 'r');
>> s = fscanf( fi1, '%s');      →   lee una cadena de caracteres

>> fi2 = fopen('entrada1.txt', 'r');
>> s = fscanf( fi2, '%d');      →   lee todos enteros posibles en el archivo

>> fi2 = fopen('entrada1.txt', 'r‘)
>> s = fscanf( fi2, '%d‘,1);    →   lee un entero
```

Instrucciones de entrada (lectura) y de salida (escritura) (cont.)

- Funciones fscanf y fprintf (cont.)

  Forma general de la función fprintf:

  count = fprintf(fi, format, var, …)

  Dirige su salida formateada hacia el archivo indicado por el identificador **fi**, **format** contiene los formatos de escritura y **count** retorna el número de bytes escritos satisfactoriamente.

  Ejemplos:

  >> fi1 = fopen(‘salida1’, ‘w’);
  >> count = fprintf(fi1, ‘el número de ecuaciones es %d \n’, n);

  → escribe el texto entre comillas y el valor de la variable n según el formato indicado (número entero)

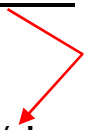
  >> fi2 = fopen(‘salida2’, ‘w’);
  >> count = fprintf(fi2, ‘el determinante es %10.4f \n’, n);

  Obs. \n en el formato obliga a crear una línea nueva al final del texto.

Instrucción de lectura “textread”

Lee datos numéricos (sin formato) o heterogéneos (con formato) de un archivo de texto (ascii)

Sintaxis sin usar formato:



a = textread(archivo)

a = textread(archivo, ‘ ', n)

a = textread(archivo, ‘ ', param, valor, ...)

a = textread(archivo, ‘ ', n, param, valor, ...)

Lee **n** líneas de **archivo** y las almacena en la variable **a**. Se supone que **archivo** contiene sólo datos numéricos.

Opciones para **param**

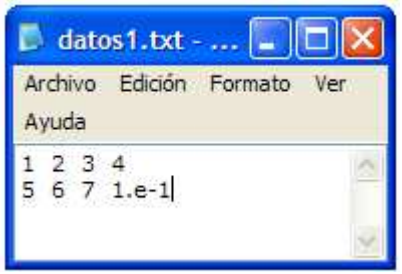
- 'delimiter': identifica el carácter de separación entre los datos
- 'headerlines': número de líneas de cabecera del archivo, estas líneas son ignoradas en la lectura
- 'commentstyle‘: con valor 'matlab‘, indica que los caracteres después de % son ignorados

Instrucción de lectura “textread” (cont.)

Ejemplos:

>> a = textread('datos1.txt ')

→ lee todas las líneas del archivo

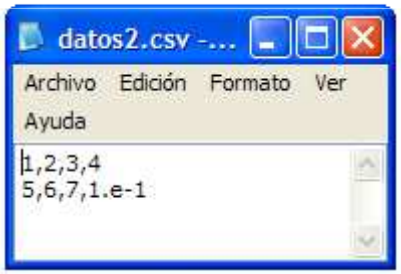


>> a = textread('datos1.txt ', ' ', 1)

→ lee la primera línea del archivo

>> a = textread('datos2.csv ', ' ', 'delimiter', ',')

→ lee todas las líneas del archivo



>> a = textread('datos2.csv', ' ', 1, 'delimiter', ',')

→ lee la primera línea del archivo

>> a = textread('datos22.csv', ' ', 'delimiter', ',', 'headerlines', 1)

→ lee todas las líneas del archivo después del encabezado

>> a = textread('datos23.csv', ' ', 'delimiter', ',', 'headerlines', 1, 'commentstyle', 'matlab') → lee todas las líneas del archivo después del encabezado, ignorando las comentadas

Instrucción de lectura “textread” (cont.)

Sintaxis usando formato:

[a, b, c, …] = textread(archivo, formato)

[a, b, c, …] = textread(archivo, formato, n)

[a, b, c, …] = textread(archivo, formato, param, valor, ...)

[a, b, c, …] = textread(archivo, formato, n, param, valor, ...)

Lee **n** líneas de **archivo** con **formato** especificado y las almacena en las variables **a**, **b**, **c**, etc., respectivamente. Los datos en **archivo** pueden ser heterogéneos (números y caracteres), pero se espera que estén organizados homogéneamente por columnas.

- El tipo de dato de cada columna se suministra en **formato** y corresponde a cada variable **a**, **b**, **c**, …
- Si **n** no se especifica o si es -1, lee el archivo completo, en caso contrario lee **n** líneas

Formatos:

%n : números reales o enteros

%d : números enteros

%f : números reales

%s : cadena de caracteres

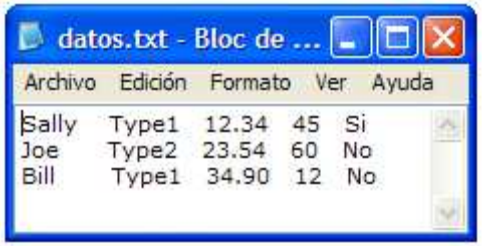
%5c : 5 caracteres (incluye espacios en blanco)

Instrucción de lectura “textread” (cont.)

Ejemplos:

```
>> [nombre, tipo, x, y, respuesta] = …
        textread('datos.txt', '%s %s %f %d %s')
```

→ lee todas las líneas del archivo, según el formato especificado y las almacena en las variables indicadas




```
nombre =
    'Sally'
    'Joe'
    'Bill'
tipo =
    'Type1'
    'Type2'
    'Type1'
x =
   12.3400
   23.5400
   34.9000
y =
    45
    60
    12
respuesta =
    'Si'
    'No'
    'No'
```

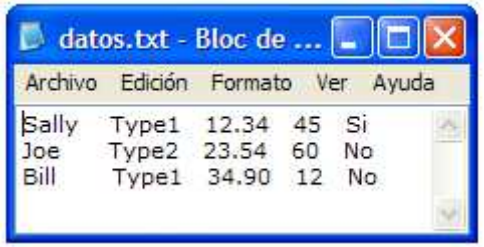
```
>> nombre = textread('datos.txt', '%s %*[^\n]')
```

→ lee la primera columna

```
nombre =
    'Sally'
    'Joe'
    'Bill'
```

Obs. Las variables nombre, tipo y respuesta son arreglos de cadena de caracteres

Instrucciones de lectura “textread” (cont.)




```
>> inicial = textread('datos.txt', '%c %*[^\n]')
```

→ lee el primer caracter

```
inicial =
   S
   J
   B
```

```
>> x = textread('datos.txt', '%*s %*s %f %*d %*s')
```

→ lee la tercera columna

```
x =
   12.3400
   23.5400
   34.9000
```

```
>> a = textread('datos.txt', '%s', 'delimiter', '\n')
```

→ cada línea como cadena de caracteres

```
a =
    'Sally    Type1   12.34   45   Si'
    'Joe      Type2   23.54   60   No'
    'Bill     Type1   34.90   12   No'
```

arreglo de cadena de caracteres

Obs.

Un arreglo de cadena de caracteres es como un vector cuyos elementos son cadenas de caracteres. Este puede ser creado usando llaves, así

>> s1 = {‘hola’ ‘si’ ‘adios’} → s1 =

'hola' 'si' 'adios'

>> ss = {'saul' 'pedro' ; 'pepe' 'maria'}

→ ss =

'saul' 'pedro‘

'pepe' 'maria'

Si construimos la cadena de caracteres concatenados verticalmente

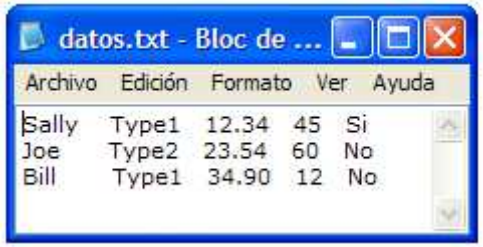
>> s2 = strvcat(‘hola', 'si', ‘adios')

La podemos interpretar como “arreglo de cadena de caracteres”, pero es no es tal.

Se tiene que s1 y s2 son diferentes, pero se puede pasar de un tipo al otro usando las funciones “char” y “cellstr”.

Además existen las funciones “ischar” y “iscellstr” para identificar entre cada uno de ellos.

Obs. (cont.)



Dado el arreglo de cadena de caracteres nombre:

>> nombre

nombre =
'Sally'
'Joe'
'Bill'

Podemos agregar una cadena de caracteres a este arreglo:

>> nombre = cellstr(strvcat(char(nombre),‘Pedro'))

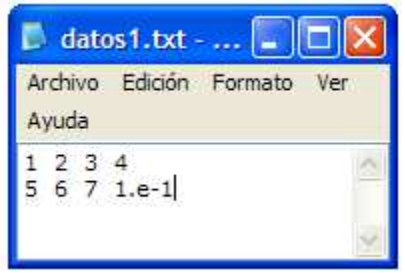
→ agrega al arreglo de cadena de caracteres ‘nombre’ el elemento ‘Pedro’

nombre =
'Sally'
'Joe'
'Bill'
‘Pedro'

>> size(nombre) → ans = 4 1

Instrucción de lectura “load”

Lee el contenido de un “archivo” y lo deposita en un arreglo con nombre “archivo”.



>> load datos1.txt

→ almacena el contenido en la variable **datos1**

la instrucción es equivalente a load –ascii datos1.txt

>> s = load(‘datos1.txt’)

→ almacena el contenido en la variable **s**

>> var = ‘datos1.txt’; s = load(var) → almacena en la variable **s**

Obs:

- El archivo puede tener cualquier extensión diferente a “.mat”, el archivo es tratado como ascii
- Es importante que todas las filas tengan el mismo número de columnas
- Las separaciones entre elementos: blancos o comas (,)
- El archivo debe tener sólo números

Instrucción de salida “save”

Guarda todas las variables especificadas del espacio de trabajo en un archivo usando formato binario de MATLAB o ascii.

Sintaxis: save(nombre del archivo, variables, formato)

Para especificar formato binario se escribe '-mat' y para formato ascii (texto) se escribe '-ascii'.

Ejemplo:

```
>> archivo = 'pqfile.mat';
>> p = rand(2, 10);
>> q = ones(10);
>> save(archivo, 'p', 'q');
```

Guarda las variables p y q en el archivo de nombre 'pqfile.mat' usando formato binario de MATLAB. Es equivalente a save(archivo, 'p', 'q', '-mat');

```
>> save(archivo, 'p', 'q', '-ascii' );
```

Guarda las variables p y q en formato ascii

Obs. Usando la instrucción load(archivo) se recuperan las variables p y q en el ambiente de trabajo.

- Funciones sscanf y sprintf

  Estas funciones permiten leer y escribir en variables tipo cadena de caracteres usando formato.

  Forma general de la función sscanf:

  [var, count, mensaje] = sscanf(s, format, size)

  Lee de la variable s de acuerdo a un formato especificado por “format” y lo devuelve en la variable var (real, vector o matriz), donde “size” (opcional) indica el número de elementos a ser leídos.
  “count” (opcional) devuelve el número de elementos leídos satisfactoriamente. “mensaje” (opcional) contendrá una cadena de caracteres indicado un mensaje de error si esto sucede o una matriz vacía en caso contrario.

  ```
  >> s = '8.19   7.315';
  >> a = sscanf(s, '%f', 2);
  ```
  a es un vector de 2 elementos con valores 8.1900 7.3150

- Funciones sscanf y sprintf (cont.)

  Forma general de la función sprintf:

  [var, mensaje] = sprintf(format, var1, …)

  Dirige su salida formateada hacia la variable tipo caracter var, de las variables var1, …, donde “format” contiene el formato de escritura.
  En este caso “mensaje” (opcional) contendrá una cadena de caracteres indicado un mensaje de error si esto sucede o una matriz vacía en caso contrario.

  ```
  >> n = 5.1
  >> resultado = sprintf('El cuadrado de %f es %12.4f \n', n, n*n);
  ```

  resultado es la cadena de caracteres:

  El cuadrado de 5.100000 es 26.0100