1.4 Ficheros m. Entrada y salida de datos por consola.

- **Ficheros m**

Hasta el momento todas las órdenes del lenguaje M las hemos ejecutado desde la ventana de comandos. De esta manera obtenemos una respuesta inmediata a la instrucción.

Sin embargo si queremos realizar un proceso formado por una secuencia de instrucciones M que vamos tecleando una a una en la ventana de comandos:

- ¿qué ocurre si cometemos una imprecisión en una instrucción a lo largo del proceso? Habría que volver a realizarlo de nuevo, orden a orden, escribiendo esta vez sin cometer errores.
- ¿qué ocurre si se quiere volver a repetir este proceso en el futuro? Habría que volver a realizarlo de nuevo, ya que las instrucciones de la ventana de comandos no se pueden guardar como tal.

Estamos de acuerdo que esta forma de actuar no es eficiente. Para resolver estos problemas existen los ficheros M. Estos ficheros no son más que archivos de texto en los que se escriben las instrucciones, en lugar de en la ventana de comandos, que quedan guardados de forma permanente, y después pueden ser ejecutados por MATLAB u Octave.

Los programas en M se escriben en ficheros de extensión **.m** (ficheros M) con un editor especial situado dentro del entorno de MATLAB u Octave, aunque es válido cualquier editor de texto no formateado, almacenándose en disco con un nombre cualquiera, por ejemplo, *programa.m*.

Para ejecutar el programa desde la ventana de comandos simplemente se debe teclear el nombre del fichero M que contiene el programa (sin la extensión). En el caso anterior sería:

```
>> programa
```

Entonces, se van ejecutando de manera secuencial todas las órdenes escritas en el fichero M, obteniendo los resultados en la ventana de comandos.

A modo de ejemplo, el alumno puede probar a teclear una a una las siguientes instrucciones en la ventana de comandos:

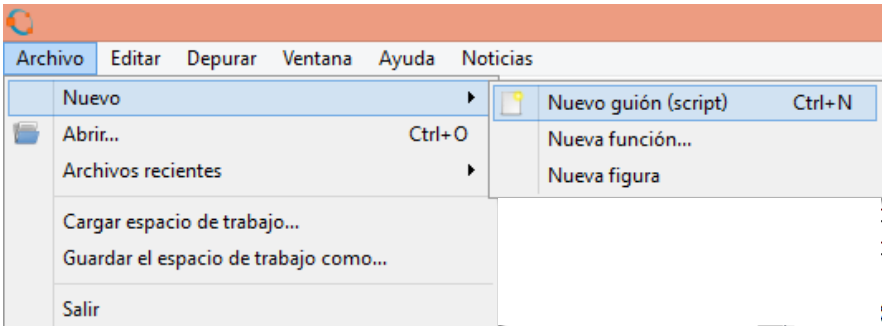
```
>>A=3

>>c=A^3

>>B=A+c
```

Imaginemos ahora que la variable A se ha tecleado incorrectamente, ya que queríamos escribir

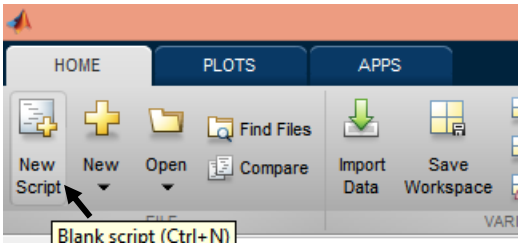
```
>> A=13
```

¿Cómo se resolvería esto desde la ventana de comandos? Evidentemente habría que volver a teclear las tres órdenes.

Probemos ahora a generar un archivo .m. Por ejemplo, desde Octave, siguiendo esta secuencia:



O desde MATLAB 2016:



Escribamos las tres instrucciones en él:

```
A=3

c=A^3

B=A+c
```

y guardémoslo como *ejemplo1.m*

Se ejecuta desde la ventana de comandos:

>> ejemplo1

obteniéndose el resultado.

En este caso la corrección del error en la variable A se resolvería corrigiendo el valor correspondiente en el fichero y volviendo a ejecutar éste. Además queda grabado en disco, luego puede ejecutarse en cualquier otro momento.

Podemos decir que hemos realizado nuestro primer programa M.

A partir de este momento se trabajará con programas, es decir escribiendo las instrucciones en ficheros M. Poco a poco se irá enriqueciendo el contenido de estos programas.

El objetivo del resto de este tema es conseguir que nuestros programas puedan tomar, en tiempo de ejecución, valores que introduce el usuario por teclado (entrada de datos) y ofrecer respuestas a través de la ventana de comandos.

- **Entrada de datos por teclado**

La función `input` permite introducir datos en un programa cuando está en modo de ejecución. La utilización es como sigue:

```
v=input('Texto de petición del dato')
```

`input` realiza dos tareas:

1) hace que aparezca en pantalla la cadena de caracteres que lleva como argumento.
2) espera a que se tecleen los datos como respuesta al texto y los memoriza en la variable `v`.

`input` no permite disponer varias variables a la izquierda de la asignación, luego todos los datos que se introduzcan deben constituir una única variable con sintaxis matricial. Ejemplo:

```
>> P=input('Introduce el radio del círculo: ')

Introduce el radio del círculo: 1.47

P =

     1.4700
```

Si como respuesta a `input` se quiere introducir una cadena de caracteres, el usuario del programa debe introducirla entre apóstrofos. El olvido de éstos provoca el fallo del programa.

```
>> Nombre=input('Introduce tu nombre y apellidos: ')

Introduce tu nombre y apellidos: Pedro Perez

error: 'Pedro' undefined near line 1 column 1
```

Para prevenirlo, se usa la función `input` con un segundo argumento `'s'` que hace que el dato introducido se tome como una cadena de caracteres sin necesidad de delimitarlo por los apóstrofos.

```
>>Nombre=input('Introduce tu nombre y apellidos: ','s')

Introduce tu nombre y apellidos: Juan Pérez González

Nombre =

Juan Pérez González

%La variable Nombre contiene una cadena de caracteres
```

A veces un programa puede fallar porque el usuario no responda a la petición del dato y pulse simplemente la tecla *enter*. Para detectarlo puede ser de utilidad la función `isempty` que devuelve cierto si la variable que tiene como argumento está vacía. Véase el ejemplo en el módulo 3 -comandos repetitivos-.

- **Salida de datos por pantalla**

Para que un programa en modo de ejecución pueda escribir textos por pantalla, se puede utilizar la función:

```
disp('Mensaje')
```

que escribe en pantalla la cadena de caracteres que tiene como argumento.

Para escribir el valor de una variable, se utiliza de esta forma:

```
disp(v)
```

que muestra en pantalla el valor de la variable `v`.

Ejemplos:

```
>> z=2;

>> disp(z)

     2

>> disp('Escritura en pantalla')

Escritura en pantalla
```

Con la función `disp` sólo se puede escribir una cadena de caracteres o una matriz (vector o escalar). Además el lenguaje M lo escribirá con formato libre, es decir, el usuario no puede seleccionar otro formato, ni realizar la escritura de una combinación de texto y datos. Para poder realizar estas acciones, se dispone de la función `fprintf`.

La función `fprintf` escribe en pantalla una combinación de datos y/o texto con el formato elegido por el programador.

Veamos la utilización general de la función `fprintf`

Para la escritura solo de texto se utiliza de igual manera que `disp`, aunque en este caso se debe terminar el texto con un salto de línea `\n`:

```
fprintf('texto\n')
```

Para escritura de texto y datos en pantalla, o sólo datos, se utiliza la función con la siguiente sintaxis

```
fprintf('formato\n', variables)
```

donde:

`variables` será la lista de variables a escribir

`formato` será la especificación del formato de escritura de las variables así como el texto que se quiera intercalar entre ellas. Además, también se podrán insertar los siguientes caracteres de control, entre otros:

`\n`: salto de línea

`\r`: retorno de carro al comienzo de la línea

`\t` tabulación horizontal

`\b`: espacio hacia atrás

Los formatos más utilizados para escribir variables son:

`%d` : adecuado para datos enteros, lógicos y para reales con decimales igual a cero. Escribe el dato como un entero. Si el dato a escribir es un real con su parte decimal no nula, no se trunca el número, se escribe con sus decimales y en formato de punto flotante.

`%f` : escribe cualquier dato numérico como un real con 6 decimales

`%s` : escribe cadenas de caracteres como tal. Si se emplean los formatos anteriores para escribir una cadena se imprimen todos los códigos ASCII de los caracteres que la forman.

Para clarificar lo explicado se ofrecen a continuación variados ejemplos:

```
>> a=2;

>> fprintf('El dato es %d\n',a)

El dato es 2

>> fprintf('%f\n',a);

2.000000

>> b=3.2;

>> fprintf('%d\n',b);

3.200000e+000
```

En el último ejemplo, el dato no es escrito como un entero ya que para ello, se debería truncar su valor. Es escrito en formato de punto flotante. Veamos otros ejemplos:

```
>> fprintf('los resultados son %d y %f\n',a,b);

los resultados son 2 y 3.200000



>> fprintf('Dato 1: %d\nDato 2: %f\n',a,b);
```

```
Dato 1: 2

Dato 2: 3.200000


>> fprintf('%s','Error en el programa');

Error en el programa
```

Cuando no se especifiquen formatos suficientes para la escritura de todos los datos, el formato se reutiliza desde el principio las veces necesarias. Por ejemplo, a continuación se pretende escribir en pantalla el valor de tres variables, sin embargo solo aparecen dos formatos; obsérvese cómo se reutiliza el formato desde el principio hasta justo antes del que no se necesita, para poder escribir el tercer dato.

```
>>a=3; b=2; c=7.3;

>>fprintf('los resultados son %d y %d\n',a,b,c);

los resultados son 3 y 2

los resultados son 7.300000e+00 y >>
```

- **Modificadores de formato en escritura de datos**

Cuando se escriben datos en pantalla con la función `fprintf`, se puede modificar el aspecto por defecto con el que se muestran mediante el uso de modificadores de formato. Su utilización se indica a continuación:

– Para datos escritos con formato `%f`, se puede incluir la siguiente información adicional en el formato:

```
%-n.mf
```

siendo:

`n`, el número mínimo de espacios utilizados en la escritura (anchura mínima de campo).

`m`, el número de decimales con los que se escribirá el dato.

signo: si se incluye el signo negativo se obliga a la justificación izquierda en el campo; si no se escribe, la justificación es a la derecha.

Veamos unos ejemplos de utilización en los que, para clarificar el formato impreso, se ha usado el símbolo ␣ para identificar el espacio en blanco.

```
>>x=56.45;

>>fprintf('%7.3f es el valor calculado\n',x)

 56.450 es el valor calculado


>>fprintf('%10.1f es el valor calculado\n',x)

      56.5 es el valor calculado


>>fprintf('%-10.1f es el valor calculado\n',x)

56.5      es el valor calculado
```

– Para datos escritos con formato %d, también se puede añadir la misma información adicional en el formato:

```
% -n.m f
```

pero en este caso, `m` no puede indicar el número de decimales sino que fija el número mínimo de dígitos a escribir. Ejemplos:

```
>>y=5;

>> fprintf('%-10.5d es el valor calculado\n',y);

00005     es el valor calculado


>>y=236;

>> fprintf('%6.2d es el valor calculado\n',y);

   236 es el valor calculado


>>y=236;

>> solucion=sprintf('%6.2d es el valor calculado',y)

solucion=

   236 es el valor calculado
```

– Para datos escritos con %s

Utilizaremos la anchura de campo y el signo de manera similar a los casos anteriores.

```
>>fprintf('%7s\n','Hola','Mundo')

   Hola

  Mundo
```