**DEPURADOR de programas MATLAB**

-        Ayuda a identificar errores de programación, tanto en programas como en funciones.

-        Permite visualizar los contenidos de la memoria de trabajo durante la ejecución del programa o la función y ejecutar el código del archivo-M línea a línea

La depuración (*debugging*) es el proceso mediante el cual aislamos y determinamos cualquier error existente en el código.

Ayuda a corregir dos tipos de errores:

1.     **Errores de sintaxis**: escribir mal el nombre de la función, olvidarnos de paréntesis, MATLAB detecta la mayoría de los errores de sintaxis y muestra un mensaje describiendo el error y mostrando su nº de línea en el archivo-M. Normalmente se pueden corregir los errores de sintaxis fácilmente basándonos en los mensajes de error de MATLAB.

2.     **Errores de ejecución y/o lógicos**: Se producen por mala codificación del algoritmo, como modificar una variable equivocada o realizar un cálculo incorrecto.

Estos errores se hacen patentes cuando el archivo-M produce resultados erróneos.

**Técnicas para aislar las causas de los errores de ejecución:**

-        Eliminar los puntos y coma de las sentencias del archivo-M. De este modo se pueden ir analizando los valores que toman las distintas variables durante la ejecución del programa. En estos casos en ocasiones será conveniente incluir la sentencia pause, al objeto de poder analizar los valores correspondientes.

-        Utilizar el depurador de MATLAB.

El depurador es útil para corregir errores de ejecución porque nos facilita el acceso a las memorias de trabajo de las funciones y a examinar o cambiar los valores que contienen.

Nos permite poner y quitar puntos de ruptura (*breakpoints*), en las líneas del programa, que permiten detener el mismo, analizar los valores de las variables y continuar con la ejecución paso a paso.

**Ejemplo**: Vamos a crear un archivo-M llamado primo.m que indica si un número pedido por teclado es primo o no. Guardaremos dicho archivo en nuestra carpeta actual.

n=input('Digame el número: ');

div=0;

k=2;

while k<=n-1

if n/k == fix(n/k)

di=div+1;

end

k=k+1;

end

if div==0

sprintf('%d es primo', n)

else

sprintf('%d NO es primo', n)

end

Ejecutaremos el programa para ver si opera de forma correcta:

EDU» primo

Digame el número: 4

ans =

4 es primo

La respuesta es incorrecta, por lo que usaremos el depurador para aislar el error en los archivos-M.

**Depuración en el PC**

Para comenzar la depuración es necesario abrir el programa en el editor de MATLAB.

Los iconos para depuración en la barra de herramientas son:

|  |  |
| --- | --- |
| **Botón de la barra de herramientas** | **Descripción** |
|  | **Set/Clear breakpoint**: fija o elimina un punto de ruptura en la línea donde está situado el cursor. |
|  | **Clear all breakpoints**: Elimina todos los puntos de ruptura que están activos. |
|  | **Step in**: ejecuta la línea actual el archivo-M y si la línea es una llamada a otra función, hace step en esa función. |
|  | **Single step**: ejecuta la línea actual del archivo-M. |
|  | **Continue**: continúa la ejecución del archivo-M hasta que finalice o hasta que encuentre otro punto de ruptura. |
|  | **Quit Debugging**: sale del estado de depuración. |

Si pulsamos el botón derecho del ratón sobre la ventana del Editor/Depurador saldrá el menú contextual con algunas de estas opciones. Las mismas opciones de la barra de herramientas están disponibles desde el menú **Debug** de la ventana del Editor/Depurador.

**Fijar puntos de ruptura**

La mayoría de las sesiones de depuración comienzan fijando un punto de ruptura en alguna/s de las líneas del programa. Estos puntos permiten parar la ejecución del programa en dichas líneas, lo que permite ver o cambiar los valores en la memoria de trabajo de la función antes de continuar la ejecución. Para fijar un punto de ruptura se debe situar el cursor en la línea correspondiente y pulsar el botón SET/CLEAR BREAKPOINT, o bien hacerlo a través del menú contextual. Con ello aparecerá un símbolo () a la izquierda de dicha línea, lo que indica que el punto de ruptura está activado. Si la línea seleccionada para situar el punto de ruptura no es una línea ejecutable válida, entonces el punto de ruptura se sitúa en la siguiente línea ejecutable.

El menú **Debug** del editor también nos permite parar la ejecución del archivo M si el código genera un aviso (warning), un error, o valores NaN (not a number, indeterminación) o Inf (infinito).

Al comienzo de la sesión de depuración, no estamos seguros de dónde está el error, por lo que pondremos el punto de ruptura en un lugar donde nos parezca oportuno para ir acotando dicho error. Situaremos el punto de ruptura en la sentencia de la línea 11:

if div==0

El número de línea está indicado en la parte inferior derecha de la barra de estado. Fijamos el punto de ruptura posicionando el cursor en la línea de texto y haciendo click sobre el botón "**Set Breakpoint**" de la barra de herramientas. Esto también puede hacerse desde el menú **Debug**, **Set Breakpoint** o sacando el menú contextual y eligiendo **Set/Clear Breakpoint**.

**Análisis de las variables**

Para llegar al punto de ruptura y analizar los valores de las variables de interés, en primer lugar debemos **ejecutar** el programa desde la ventana de comandos de MATLAB:

primo

introduciendo el valor 4 nuevamente. Cuando la ejecución del archivo‑M alcanza el punto de ruptura, se muestra en la ventana del editor una flecha amarilla () a la izquierda de la próxima sentencia en ejecutarse. Cuando aparece una flecha amarilla hacia abajo () a la izquierda del texto significa que se ha alcanzado el final del programa o función. Esto nos permite examinar las variables antes de volver a la función de llamada.

Analizaremos el valor de la variable div (obteniendo div=0), lo que nos indica que el error se ha producido antes, ya que para n=4 dicha variable debiera tener el valor 1, al ser divisible 4 entre 2.

Ya que el error se encuentra en alguna sentencia anterior, finalizaremos esta sesión de depurado (opción Debug+Quit Debugging), para colocar un nuevo punto de ruptura en alguna línea anterior.

**Continuación y ejecución paso a paso**

Eliminaremos el punto de ruptura de la línea 11 situándonos sobre esa línea y eligiendo las opciones de menú **Debug**, **Clear Breakpoint** o sacando el menú contextual y eligiendo **Clear Breakpoint** o pulsando el botón **Set/Clear Breakpoint** de la barra de herramientas.

Situamos ahora un nuevo punto de ruptura en la línea 5, y volvemos a repetir la ejecución del programa con el valor n=4. De esta manera el programa se detendrá en la evaluación del mientras, de modo que podremos ejecutar paso a paso todas las iteraciones del lazo. Para ello, pulsaremos el botón SINGLE STEP, que nos llevará a la línea 6. En la situación actual, n=4 y k=6, se cumplirá la condición del if, con lo cual al ejecutar la sentencia nuevamente con **SINGLE STEP** pasaremos a la línea 7. De este modo se almacena el valor 1 en la variable de la izquierda de la asignación. Si continuamos ejecutando paso a paso las sentencias, vemos que para el siguiente valor del lazo k=3 no se cumple la condición del if, con lo que las variables quedan con el mismo valor que tenían antes. Para el valor k=4 no se repite ya el lazo mientras, por lo que el programa saltará a la línea 11. En esta situación vemos que la condición de dicha línea se va a cumplir, ya que div=0, pero si evaluamos la variable de la izquierda de asignación de la línea 7 vemos que vale 1. En esta situación nos damos cuenta rápidamente que el error está en los nombres de las variables, ya que en la línea 7 le hemos llamado **div**, mientras que en la línea 11 se llama **div**, cuando debiera llamarse igual en ambos sitios.

 <![endif]>

**Fin de la sesión de Depuración**

Antes de modificar el programa para corregir el error, debemos abandonar la opción de depuración con **DEBUG+QUIT DEBUGGING** (o bien con el botón **QUIT DEBUGGING** de la barra de herramientas). Seguidamente corregimos el error, guardamos el programa y podemos comprobar que tras la corrección ya opera correctamente. Antes de volver a ejecutarlo, recordad que debéis eliminar el punto de ruptura, pues de no hacerlo nuevamente entraremos en un proceso de depuración.

Otras alternativas para finalizar la sesión de depuración son seleccionar la opción de menú **File**, **Exit Editor/Debugger** , o bien cerrar la ventana directamente con el botón Cerrar ventana de Windows.

**Programas que contengan funciones**

Para depurar programas que contengan funciones definidas por el usuario se procede de igual modo, salvo que al llegar a la sentencia que contenga la llamada a la función tenemos dos opciones:

* si queremos depurar también la función, debemos ejecutar la sentencia utilizando el botón botón STEP-IN en lugar del SINGLE STEP. De este modo el depurador abrirá el fichero que contenga la función y permitirá depurarla paso a paso. Cuando el depurador "salte" al fichero de la función podemos seguir depurándola de la forma habitual.
* si no queremos depurar la función, sino que nos interesa seguir depurando el programa, ejecutaremos la sentencia usando, como siempre, el botón SINGLE STEP, con lo que podremos ver el resultado de la ejecución de la función, pero sin analizarlo paso a paso. Esto es recomendable cuando estemos seguros que la función opera correctamente.