

Unit 8: Handling Errors

Objectives

- Understand the importance of making robust programs which handle the errors correctly.
- Understand the concept of exception.
- Understand the use of try-catch blocks to capture and handle exceptions.

Handling Errors and Exceptions

- Regardless how carefully you write a program, it may not always produce the expected results when executed under different conditions.

Handling Errors and Exceptions

- Regardless how carefully you write a program, it may not always produce the expected results when executed under different conditions.
- Different types of errors:
 - ❑ The program stops abnormally, and you get an error message from MATLAB
 - ADVICE: Read the error message...
 - ❑ The program never stops
 - Press the keys Ctrl and C at the same time to stop the execution
 - ❑ The result of the program is not correct

Debugging

- Debug: process of finding errors (“bugs”) in a computer program
- How to debug a program:
 - MATLAB Debugger:
 - You can execute the program line by line, checking the current values of the variables and confirming that the execution path is the one you expected
 - Manually:
 - Include messages in code to confirm the path of execution
 - Print on screen the values of the variables
 - Comment some parts of the program to isolate the area of the error

Debugging. Example

```
clear;  
cont = 0;  
menu = input('Introduce the type of menu: ','s');  
sdish = input('Introduce the name of the dish','s');  
while (isempty(sdish) == 0)  
    cont = cont + 1;  
    menu{cont} = sdish;  
    sdish = input('Introduce the name of the dish','s');  
end;
```

Error: ??? Cell contents assignment to a non-cell array object.

Debugging. Example

TRY TO ISOLATE THE
AREA OF THE ERROR

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the dish','s');
while (isempty(sdish) == 0)
    cont = cont + 1;
    menu{cont} = sdish;
    sdish = input('Introduce the name of the dish','s');
end;
```

Error: ??? Cell contents assignment to a non-cell array object.

Debugging commenting areas

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the dish','s');
% while (isempty(sdish) ==0)
%   cont = cont +1;
%   menu{cont} = sdish;
%   sdish = input('Introduce the name of the dish','s');
% end;
```


Debugging commenting areas

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the dish','s');
% while (isempty(sdish) ==0)
%   cont = cont +1;
%   menu{cont} = sdish;
%   sdish = input('Introduce the name of the dish','s');
% end;
```

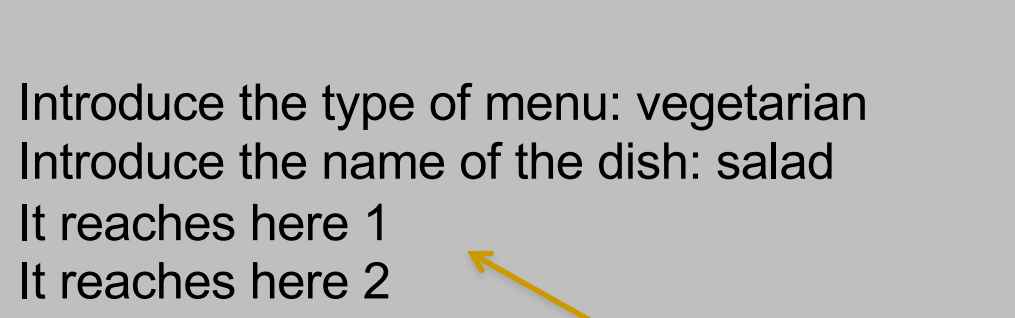
I don't get the error now... So then error must be in the while

Debugging displaying messages

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the dish','s');
while (isempty(sdish) == 0)
    disp('It reaches here 1');
    cont = cont + 1;
    disp('It reaches here 2');
    menu{cont} = sdish;
    disp('It reaches here 3');
    sdish = input('Introduce the name of the dish','s');
    disp('It reaches here 4');
end;
```

Debugging displaying messages

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the ingredient','s');
while (isempty(sdish) == 0)
    disp('It reaches here 1');
    cont = cont + 1;
    disp('It reaches here 2');
    menu{cont} = input('Introduce the name of the dish: ','s');
    disp('It reaches here 3');
    slng = input('Introduce the name of the dish','s');
    disp('It reaches here 4');
end;
```



Introduce the type of menu: vegetarian
Introduce the name of the dish: salad
It reaches here 1
It reaches here 2

The error is between the
“reaches here 2” and
“reaches here 3”

Debugging displaying messages

```
clear;
cont = 0;
menu = input('Introduce the type of menu: ','s');
sdish = input('Introduce the name of the ingredient','s');
while (isempty(sdish) == 0)
    disp('It reaches here 1');
    cont = cont + 1;
    disp('It reaches here 2');
    menu{cont} = sdish;
    disp('It reaches here 3');
    sling = input('Introduce the name of the dish','s');
    disp('It reaches here 4');
end;
```

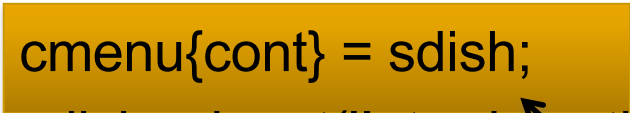
Debugging. Example

```
clear;  
cont = 0;  
menu = input('Introduce the type of menu: ','s');  
sdish = input('Introduce the name of the dish','s');  
while (isempty(sdish) == 0)  
    cont = cont + 1;  
    menu{cont} = sdish;  
    sdish = input('Introduce the name of the dish','s');  
end;
```

The problem is here...

Debugging. Example

```
clear;  
cont = 0;  
smenu = input('Introduce the type of menu: ','s');  
sdish = input('Introduce the name of the dish','s');  
while (isempty(sdish) == 0)  
    cont = cont + 1;  
    cmenu{cont} = sdish;  
    sdish = input('Introduce the name of the dish','s');  
end;
```



I was using the same variable name, *menu*, for storing strings and cells. Before using a variable for cells you need to clear it!

Debugging

- With a bit of practice you will learn to identify the errors very quickly.
- Remember: you learn a lot from your errors. You don't make the same mistake twice!
- Unfortunately, we have to live with errors:
 - The user introduces wrong or unexpected data
 - The file contains wrong data
 - ...

This is why we need to create programs which produce a nice output.. even in the case that an unexpected problem occurs

Handling Errors and Exceptions

- An exception is an unexpected condition or an error that stops the normal program flow.
 - ❑ *Fatal exceptions*: the execution of the program is finalized
 - ❑ *Recoverable Exceptions*: the possibility of recovering from the error should be provided
- Examples:
 - ❑ Trying to open a file which doesn't exist
 - ❑ Divide a value by zero
 - ❑ Variable with no value assigned

Handling Errors and Exceptions

- It is important to control the execution of the program and include error checking to ensure reliable operation under all conditions.
 - Create a program structure which **prevents exception** to happen
 - Create a program structure which **correctly handle** errors
 - Try-catch
 - lasterr
 - ferror
 - ..

Program Structure

```
file = input('Introduce the file name', 's');
[mydata, verr] = readMydata(file);
if strcmp(verr,'ok')
    ...
    [rdo, verr] = computeRdo(info1, info2);
    if strcmp(verr,'ok')
        ...
    end;
end
if strcmp(verr,'ok')
    disp('Program finishes without errors');
elseif strcmp(verr,'errAccess')
    disp('Error when accessing a file');
elseif strcmp(verr,'errComputing')
    ...
end
```

Programmers like to include some extra parameters in functions in order to retrieve information about their correct execution

```
function [data, verr] = readMydata(filename)
verr = 'ok';
...
if ...
    verr = 'errAccess';
...
If ..
    verr = 'errReading';
...
end
```

```
function [rdo, verr] = computeRdo(var1, var2)
verr = 'ok';
...
if (var2 == 0) || (var1 == 0)
    verr = 'errComputing';
...
end
```

Managing Exceptions

try

Statements that might generate the exception

catch *MException*

Actions to manage the exception (examine error, attempt to recover, or clean up and abort)

end

- If no errors are encountered, MATLAB skips the catch block
- If any error of the try statements fail, MATLAB immediately exits the *try* block, leaving any remaining statements in that block unexecuted, and enters the catch block.
- When a variable name is specified in the catch statement (*ME*), the variable is used to store information about the error that can be useful in determining what happened and how to proceed.
- The *rethrow(MException)* command allow to raise the exception again, as it wouldn't been captured by the catch block.

Managing Exceptions: try-catch

Example

```
try
    fileId = fopen('myfile','r');
    dataMyFile = fread(fileId);
    fclose(fileId);
catch
    disp('Error when accessing the file');
end
```

In case the file does not exist and therefore MATLAB cannot open it, the 'catch' block will capture the exception produced when trying to read it and it will display an error message. The execution will continue after the 'end' statement.

Managing Exceptions. try-catch + var

Example

```
try
    fileId = fopen('myfile','r');
    dataMyFile = fread(fileId);
    fclose(fileId);
catch ME1
    disp('Error when accessing the file');
    fprintf('Data error: %s', ME1.identifier);
end
```

The information about the error is stored in the variable ME1. In this case the catch block will retrieve the identifier of the error and display it to the user.

Managing Exceptions. try-catch + rethrow

Example

```
try
    file = input('Introduce the file name', 's');
    data = readMydata(file);
    disp('The program finished correctly.');
```

catch

```
    disp('Program execution finished with errors');
End
```

```
function [data] = readMydata(filename)
try
    fileId = fopen(filename,'r');
    dataMyFile = fread(fileId);
catch ME1
    disp('Error when accessing the file');
    fprintf('Data error: %s', ME1.identifier);
    rethrow (EM1) ;
End
```

The information about the error is stored in the variable ME1. In this case the catch block will retrieve the identifier of the error and display to the user. Then, as the error is rethrow the catch in the main program is also activated

Managing Error and Exceptions

- Matlab provides some functions to manage errors and exceptions:
- `[message, errnum] = error(fileId)`
returns the error message for the most recent file I/O operation on the specified file
- `s = lasterror`
Last error message and related information
- ...

Bibliography

- More information about handling errors and exceptions:

<http://www.mathworks.com/help/techdoc/ref/f16-42340.html#f16-7291>