How To: make a simple text menu in MATLAB

ENGR 112
Spring 2018

The Basic Menu Algorithm

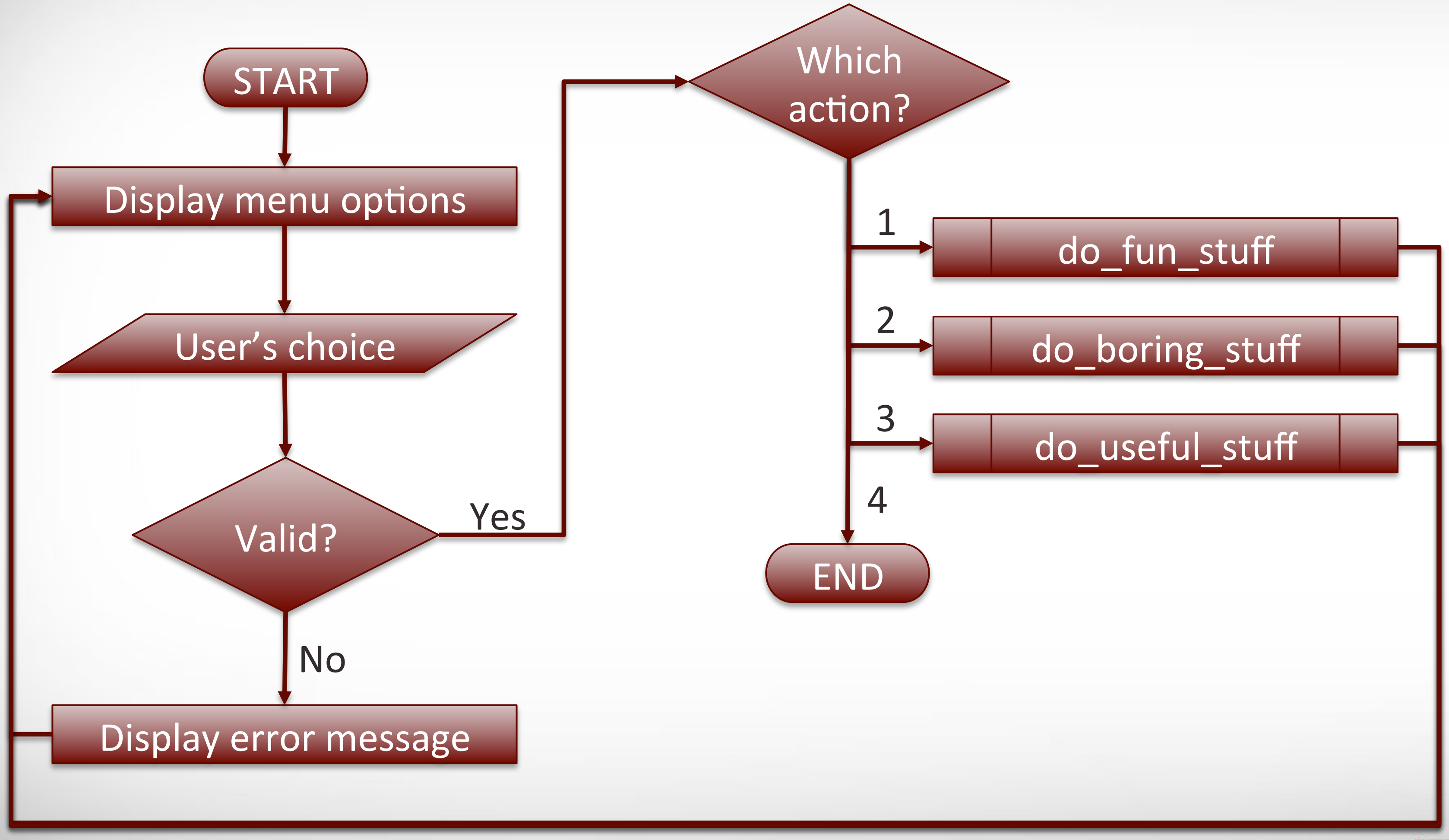
Every menu follows the same basic recipe:

```
1. Display menu options
2. Prompt user to select an option
3. Validate user input
4. Perform an action based on user input
5. GOTO 1
```

Example Menu

```
What do you want to do?
1) Do fun stuff
2) Do boring stuff
3) Do useful stuff
4) Exit
Enter your choice:
```

Program Flow



Abstraction

The value of a text menu is that it simplifies complex operations by presenting the user with a natural language interface. All of the complexity is hidden behind the interface.

The menu should be formatted so that it is easy to read. The options should be clear and concise.

Input validation

The user should be regarded as if they are malicious. The user will try to make the program crash, have errors, or behave strangely. The code must be written in such a way that the user has a very hard time doing this.

Valid input is input which will not cause the program to crash, have errors, or behave strangely.

Only valid input should be accepted. Invalid input should be detected and handled gracefully.

The menu drives the program

When the user makes a valid choice, the program should carry out the action corresponding to that choice.

If user picks option 3, then “do useful stuff”.

After the action is complete, the user is returned to the menu to make another choice.

This continues until the user chooses to exit, at which point the program will stop running.

Menu options → actions

Each option in the menu corresponds to an action defined by the program's authors.

- Execute a script
- Invoke a function
- Set the value of a variable
- End the program

The example has 4 actions: do fun stuff, do boring stuff, do useful stuff, exit

Modularity

Unless an action is very simple (i.e. only 2 or 3 lines of code), it is best to put the code which defines that action into another script or function which can be invoked by the menu code. This helps to organize the code and make it more readable and manageable.

This example will use 3 functions which carry out the operations required to do fun, boring, and useful things.

Implementation

The structure of the program looks like this:

```
set done = false
while not done
  print the menu options
  prompt for user input
  if input is valid then
    carry out chosen action
  else
    print error message
```

Print menu options

Printing menu options is just a sequence of print statements:

```
fprintf('What do you want to do?\n');
fprintf('1) Do fun stuff\n');
fprintf('2) Do boring stuff\n');
fprintf('3) Do useful stuff\n');
fprintf('4) Exit\n');
```

You could write a function to make this more general, so that the same code can be used to print any list of options.

Prompt for user input

Getting input from the user requires the `input` function. By default, MATLAB will try to evaluate the input expression. **That is unsafe. We do not want that**. We will force MATLAB to interpret the input as a string (preventing evaluation). To do this, we give `input` a second argument with the value `'s'`, which means "read the input as a string".

```
choice = input('Enter your choice: ','s')
```

Input validation

The menu has 4 options.

Each option is identified by a number.

Input from the user is obtained as a string.

The only valid inputs are: `‘1’`, `‘2’`, `‘3’`, and `‘4’`. NB: these are **strings**.

```
if ismember(choice,{‘1’,‘2’,‘3’,‘4’})
   % carry out correct action
else
   % display error message
end
```

Determine correct action

We use a `case` structure to determine which action to take based on the input provided by the user. Each case corresponds to a single action.

```
switch choice
   case '1'
      % do action for option 1
   case '2'
      % do action for option 2
   [and so on]
end
```

Simplify input validation

Since the `case` structure enumerates valid input, an `otherwise` clause can be used to handle the invalid inputs in a natural way.

```
switch choice
   case '1'
      % do action for option 1
   [other cases]
   otherwise
      % handle invalid input
end
```

The previous validation code is unnecessary.

Carry out the actions

In `case` 1, we call `do_fun_stuff()`

In `case` 2, we call `do_boring_stuff()`

In `case` 3, we call `do_useful_stuff()`

In `case` 4, we set `done = true`

`otherwise`, we print an error message

Functions are files

Remember: in MATLAB, each top-level function is stored in a file of the same name.

`do_fun_stuff()` is stored in `do_fun_stuff.m`

`do_boring_stuff()` is stored in `do_boring_stuff.m`

`do_useful_stuff()` is stored in `do_useful_stuff.m`

If an action is carried out by a script, that script is stored in another file and should have a useful and informative name.

Put it all together

```
continueProgram = true; % equivalent to done = false
while continueProgram
    print_menu_options; % this is a script that prints options
    choice = input('Enter your choice: ','s');
    switch choice
        case '1'
            do_fun_stuff();
        case '2'
            do_boring_stuff();
        case '3'
            do_useful_stuff();
        case '4'
            continueProgram = false;
        otherwise
            fprintf('Please choose a option between 1 and 4\n');
    end
end
```

Advanced Menus

- Fancy formatting
- Menus with hidden options
- Menus that invoke other menus
- Menus that have different options based on the current state of program

If you long for old school cool, text-based adventure games are just lots of menus.

Please send all complaints to /dev/null

Everything else can be sent to kurwitz@tamu.edu

TEXAS A&M UNIVERSITY