Control structures (conditionals and loops)

Analysis for Neuroscientists

September 26, 2022

1 Conditional Statements

An extension to *if-else-elseif* statements is *switch-case-otherwise*.

```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
    ...
   otherwise
      statements
end
```

Let's look at a couple of examples using *switch*.

```
choice = 1;
switch choice
    case 1
       x = -pi:0.01:pi;
    case 2
       x = 0:1:physconst('earthradius');
    otherwise
       % does not know anything about x
end
```

Another example:

```
% Determine which type of plot to create based on the value
of plottype. If plottype is either 'pie' or 'pie3', create a
3-D pie chart. Use a cell array* to contain both values.

 x = [12 64 24];
 plottype = 'pie3';

 switch plottype
     case 'bar'
         bar(x)
         title('Bar Graph')
```

```
    case {'pie','pie3'}
        pie3(x)
        title('Pie Chart')
    otherwise
        warning('Unexpected plot type. No plot created.')
end
```

*What is a *cell array?*
A *cell array* is a data type with indexed data containers called *cells*, where each cell can contain any type of data. Cell arrays commonly contain either lists of text, combinations of text and numbers, or numeric arrays of different sizes. Refer to sets of cells by enclosing indices in smooth parentheses, (). Access the contents of cells by indexing with curly braces, {}.

```
% When you have data to put into a cell array, create the
array using the cell array construction operator, {}.

C = {1,2,3;
     'text',rand(5,10,2),{11; 22; 33}}
```

2 Loop Control Structures

A *nested loop* is a loop within a loop, an inner loop within the body of an outer one. How this works is that the first pass of the outer loop triggers the inner loop, which executes to completion. Then the second pass of the outer loop triggers the inner loop again. This repeats until the outer loop finishes.

```
for ii = 1:50
 % do work in outer loop
 % ...

  for jj = 1:25
  % do work in inner loop
  % ...
  end

end
```

There will be multiple occasions where a nested loop will be useful. Let's look at an example.

We would like to design the game Sudoku[1]. Our input will be provided (a puzzle and the answer).

```
sudokuPuzzle = [1 1 0 0 1 0 0 0 0 ; ...
                1 0 0 1 1 1 0 0 0 ; ...
                0 1 1 0 0 0 0 1 0 ; ...
                1 0 0 0 1 0 0 0 1 ; ...
                1 0 0 1 0 1 0 0 1 ; ...
                1 0 0 0 1 0 0 0 1 ; ...
                0 1 0 0 0 0 1 1 0 ; ...
                0 0 0 1 1 1 0 0 1 ; ...
                0 0 0 0 1 0 0 1 1];

sudokuSolution = [5 3 4 6 7 8 9 1 2 ; ...
                  6 7 2 1 9 5 3 4 8 ; ...
                  1 9 8 3 4 2 5 6 7 ; ...
                  8 5 9 7 6 1 4 2 3 ; ...
                  4 2 6 8 5 3 7 9 1 ; ...
                  7 1 3 9 2 4 8 5 6 ; ...
                  9 6 1 5 3 7 2 8 4 ; ...
                  2 8 7 4 1 9 6 3 5 ; ...
                  3 4 5 2 8 6 1 7 9];
```

The first we would like to do is display the puzzle but in a stylistic manner using "character graphics."

Maybe something like the following:

```
_________________________
| 5 3 _ | _ 7 _ | _ _ _ |
| 6 _ _ | 1 9 5 | _ _ _ |
| _ 9 8 | _ _ _ | _ 6 _ |
_________________________
| 8 _ _ | _ 6 _ | _ _ 3 |
.
.
.
```

How can we do it using nested loops and conditional statements?

Let's start with a single 3x3 rectangle.

```
for ii = 1:3
    dispOutput = [];
```

[1] https://en.wikipedia.org/wiki/Sudoku

```
    for jj = 1:3
        if sudokuPuzzle(ii,jj) == 1
            dispOutput = [dispOutput ' ' num2str(sudokuAnswer(ii,jj))];
        else
            dispOutput = [dispOutput ' ' '_'];
        end
    end
    disp(dispOutput);
end%
```

Now try to expand it to the entire board using nested loops.

Let's now outline the plan for implementing the Sudoku game:

1. Input of the *Puzzle* and *Solution*
2. Display the *Puzzle*
3. Allow the user to input their solution and display the updated board
4. Check that the solution is correct

The problem set for this week will be a LiveScript that will implement the above steps and allow one to play Sudoku on MATLAB.

There are a few things that we will cover in class but that you should further read on LiveScript and Boolean Logic as it may help you solve this problem set.

How to use LiveScript controls: https://www.mathworks.com/support/search.html/videos/how-to-use-live-editor-controls-1569868241587.html

More on Boolean Logic if you are not confident about it: https://www.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html