Fundamental Programming Principles: Variables and Functions LAB

Beyond the Mouse

GEOS 436/636

Jeff Freymueller, Sep 23, 2014

"The Uncomfortable Truths Well",
http://xkcd.com/568 (April 13, 2009)

Structures

- Suppose you have a set of variables that go together semantically. A structure (or struct) lets you package them together, making it easy to keep track of things.
- A ***struct*** has one or more ***fields***, which are named, and each can store a value (or vector, or array, or a struct, or …)
- You define a struct by naming its fields and assigning a value to each (or use [] for an empty value).
- Access a field like this: `weather.year`, `weather.temp(5)`
  - Or `getfield(weather, "year")` or `getfield(weather, name)` where `name` is a variable.

Struct Example 1

```
>> load st_elias
>> st_elias

st_elias =

        name: 'St. Elias region interpolated grid'
        type: 'velocity'
        unit: 'cm/yr'
        bbox: [2x2 double]
     timedep: []
     div_lon: 0.2500
     div_lat: 0.2500
    lonarray: [41x1 double]
    latarray: [27x1 double]
        east: [27x41 double]
       north: [27x41 double]
      height: [27x41 double]
```

The commands “load” and “save” let you store workspace or variables to disk. In this case, there is a file `st_elias.mat` that stores the saved variable `st_elias`.

This struct stores a gridded data set (in this case a model computation). It has some meta-data, a bounding box (`bbox`), information about the grid, and then data values.

Struct Example 2

- Let's suppose you have a data file, and you need to keep track of the file meta-data as well as the data values. For example, a SAR image. Possible fields are:
  - amplitude, phase : arrays of data values
  - x, y : positions of each (georeferenced) pixel
  - satellite_name
  - track_num, frame_num : identify the image
  - More meta-data

Example 2 continued

- Define the structure:
  - `my_image = struct('satellite_name', [], 'track_num', [], 'frame_num', [], 'x', [], 'y', [], 'amplitude', [], 'phase', []);`
  - This is a set of label, value pairs. [] means an empty array. You could put values here, but be aware that MATLAB won't always do what you expect if some values are arrays.
    - I put in empty values for that reason.
- Now populate it with values
  - `my_image.satellite_name = 'Envisat';`
  - `my_image.track_num = 1745;`
- Access the values
  - `small_x_values = (my_image.x < -500);`

Struct Example 3

- Suppose you have a data set that has a number of arrays of the same size. There might also be some other information.
  - For each day of year, you have temperature, pressure and humidity readings
- What are the fields?
  - Year (scalar)
  - Doy (Day_of_year) (array)
  - temperature, pressure, humidity (arrays)
- There are >2 different ways you might store this data
  - Make a struct that stores fields '`year`', '`doy`', '`temp`', '`p`', '`humidity`' as arrays
  - Make a struct for each day stores fields '`year`', '`doy`', '`temp`', '`p`', '`humidity`' as scalars, and create an array of these structs

What's the Difference?

- To a large extent, it is a matter of what makes sense to you.
  - There is no unique right answer
- My practice is that one struct holds one 'data set', with everything for that data set.
- Is your logical 'data set' one day of data, or the years' worth of data?

Example 3: array of structs

- The easy way to make an array of structs is to put all the numerical values into cell arrays and use these to define the struct.
  - You'll get an error message if the cell arrays are different sizes.
  - It is possible to confuse MATLAB and get puzzling errors
    - `s = struct('type', {'big','little'},'color','red','x',{3 4})`
- This produces a 2 element array of structs. You can access the values in different ways:
  - `s(2).type`
  - `mean( [s.x] )`
- You can also add an element to an existing array
  - `s(3).x = 5`

Let's Explore MATLAB for a while

- Structs
  - Let's look at some examples.