Previous Lecture:
- File I/O, use of cell array

Today’s Lecture:
- Structures
- Structure array (i.e., an array of structures)
- A structure with array fields

Announcements:
- Discussion this and next two weeks in the computer labs (Upson and Phillips)
- Project 5 due Fri 4/22 at 11pm
- Prelim 2 on Tues 4/26 at 7:30pm. Email Randy Hess (rbh27) now if you have an exam conflict (include the course and instructor info of the conflicting exam)
Data are often related

- A point in the plane has an x coordinate and a y coordinate.
- If a program manipulates lots of points, there will be lots of x’s and y’s.
- Anticipate clutter. Is there a way to “package” the two coordinate values?
Packaging affects thinking

Our Reasoning Level:

P and Q are points. Compute the midpoint M of the connecting line segment.

Behind the scenes we do this:

\[ M_x = \frac{(P_x + Q_x)}{2} \]
\[ M_y = \frac{(P_y + Q_y)}{2} \]

We’ve seen this before: functions are used to “package” calculations.

This packaging (a type of abstraction) elevates the level of our reasoning and is critical for problem solving.
Options for storing a point (-4, 3.1)

- **Simple scalars**
  
  xdat 4  ydat 3.1

- **Simple vector**
  
  ptdat [-4 3.1]

- **Cell array**
  
  ptdatc { [-4 3.1] }

- **Struct**
  
  pt x 4  y 3.1

---

Ungrouped data

Related data grouped into an array. X-coord implicitly labelled 1; y-coord implicitly labelled 2

Related data grouped into a struct variable. Explicit, clear labelling is possible via field names
Example: a \textbf{Point} structure

\texttt{\% p1 is a Point}
\texttt{p1.x= 3;}
\texttt{p1.y= 4;}

\texttt{\% p2 is another Point}
\texttt{p2.x= -1;}
\texttt{p2.y= 7;}

\textbf{A Point has two properties—fields—x and y}
Working with Point structures

\[
p1.x=3; \quad p1.y=4;
p2.x=-1; \quad p2.y=7;
\]

% Distance between points p1 and p2
\[
D= \sqrt{(p1.x-p2.x)^2 + (p1.y-p2.y)^2);
\]

Note that \( p1.x, \ p1.y, \ p2.x, \ p2.y \) participate in the calculation as variables—because they are.
Different ways to create a structure

% Create a struct by assigning field values
p1.x = 3;
p1.y = 4;

% Create a struct with built-in function
p2 = struct('x', -1, 'y', 7);

p2 is a structure.
The structure has two fields.
Their names are \textbf{x} and \textbf{y}.
They are assigned the values -1 and 7.
Accessing the fields in a structure

\[ A = p1.x + p1.y; \]

**Syntax:**

`StructName.FieldName`
A structure can have fields of different types

\[
A = \text{struct}('sname', 'New York',
\quad 'capital', 'Albany',
\quad 'pop', 15.5)
\]

- Can have combinations of string fields and numeric fields
- Arguments are given in pairs: a field name, followed by the value
Legal/Illlegal maneuvers

\[ Q = \text{struct}('x', 5, 'y', 6) \]

\[ R = Q \quad \% \text{Legal. } R \text{ is a copy of } Q \]

\[ S = (Q+R)/2 \quad \% \text{Illegal. Must access the} \\
\quad \% \text{fields to do calculations} \]

\[ P = \text{struct}('x', 3, 'y') \quad \% \text{Illegal. Args must be} \\
\quad \% \text{in pairs (field name} \\
\quad \quad \% \text{followed by field} \\
\quad \quad \% \text{value)} \]

\[ P = \text{struct}('x', 3, 'y', []) \quad \% \text{Legal. Use [] as} \\
P.y = 4 \quad \% \text{place holder} \]
Structures in functions

function d = dist(P,Q)

% P and Q are points (structure).
% d is the distance between them.

d = sqrt((P.x-Q.x)^2 + ... 
        (P.y-Q.y)^2);

Example “Make” Function

function P = MakePoint(x,y)
% P is a point with P.x and P.y
% assigned the values x and y.
P = struct('x',x,'y',y);

a = 10;  b = rand;
P = MakePoint(a,b); % create a point struct
% according to definition
% in MakePoint function
Another function that has structure parameters

function DrawLine(P,Q,c)
% P and Q are points (structure).
% Draws a line segment connecting
% P and Q. Color is specified by c.

plot([P.x Q.x],[P.y Q.y],c)
Pick Up Sticks

\[
s = 'rgbmcy';
\]

for k=1:100

\[
\begin{align*}
P &= \text{MakePoint}(\text{randn}, \text{randn}); \\
Q &= \text{MakePoint}(\text{randn}, \text{randn}); \\
c &= s(\text{ceil}(6*\text{rand})); \\
\text{DrawLine}(P, Q, c)
\end{align*}
\]

end

Generates two random points and connect them using one of six colors chosen randomly.
Structure Arrays

- An array whose components are structures
- All the structures must be the same (have the same fields) in the array
- Example: an array of points (point structures)
An Array of Points

\[ P(1) = \text{MakePoint}(0.50, 0.86) \]
An Array of Points

\[ P(2) = \text{MakePoint}(-.50, .86) \]
An Array of Points

\[ \text{P(3)} = \text{MakePoint}(-1.0, 0.0) \]
An Array of Points

\[ P(4) = \text{MakePoint}(-0.50, -0.86) \]
An Array of Points

\[ P(5) = \text{MakePoint}(0.50, -0.86) \]
An Array of Points

\[ P(6) = \text{MakePoint}(1.0, 0.0) \]
Function returning an array of points (point structures)

```matlab
function P = CirclePoints(n)
    theta = 2*pi/n;
    for k=1:n
        c = cos(theta*k);
        s = sin(theta*k);
        P(k) = MakePoint(c,s);
    end
```
Example: all possible triangles

- Place $n$ points uniformly around the unit circle.
- Draw all possible unique triangles obtained by connecting these points 3-at-a-time.

\[(i, j, k) = (1, 2, 4)\]

\[(i, j, k) = (1, 2, 6)\]
function DrawTriangle(U,V,W,c)
% Draw c-colored triangle;
% triangle vertices are points U, V, and W.

fill([U.x V.x W.x], ... [U.y V.y W.y], c)
The following triangles are the same: (1,3,6), (1,6,3), (3,1,6), (3,6,1), (6,1,3), (6,3,1)
Bad! i, j, and k should be different, and there should be no duplicates

% Given P, an array of point structures
for i=1:n
    for j=1:n
        for k=1:n
            DrawTriangle(P(i),P(j),P(k),'m')
            pause
            DrawTriangle(P(i),P(j),P(k),'k')
        end
    end
end
All possible \((i, j, k)\) combinations but avoid duplicates. Loop index values have this relationship \(i < j < k\)

\[
\begin{align*}
i & = 1 \\
& 1 2 3 \\
& 1 2 4 \\
& 1 2 5 \\
& 1 2 6 \\
& 1 3 4 \\
& 1 3 5 \\
& 1 3 6 \\
& 1 4 5 \\
& 1 4 6 \\
& 1 5 6
\end{align*}
\]

\[
\begin{align*}
i & = 2 \\
& 2 3 4 \\
& 2 3 5 \\
& 2 3 6 \\
& 2 4 5 \\
& 2 4 6 \\
& 2 5 6
\end{align*}
\]

\[
\begin{align*}
i & = 3 \\
& 3 4 5 \\
& 3 4 6 \\
& 3 5 6
\end{align*}
\]

\[
\begin{align*}
i & = 4 \\
& 4 5 6
\end{align*}
\]

for \(i=1:n-2\)
\[
\begin{align*}
& \text{for } j=i+1:n-1 \\
& \text{for } k=j+1:n \\
& \text{disp}([i \ j \ k]) \\
& \text{end} \\
& \text{end} \\
& \text{end}
\end{align*}
\]
All possible \((i,j,k)\) combinations but avoid duplicates. Loop index values have this relationship \(i < j < k\)

Both versions print all possible, unique combinations of \((i,j,k)\), but the left fragment is far more efficient
All possible (i,j,k) combinations but avoid duplicates. Loop index values have this relationship $i < j < k$

for $i=1:n-2$
  for $j=i+1:n-1$
    for $k=j+1:n$
      \% Draw triangle with % vertices $P(i), P(j), P(k)$
      end
  end
end
All possible **unique** triangles

```matlab
% Drawing on a black background
for i=1:n-2
    for j=i+1:n-1
        for k=j+1:n
            DrawTriangle( P(i), P(j), P(k), 'm' )
            DrawPoints(P)
            pause
            DrawTriangle(P(i),P(j),P(k),'k')
        end
    end
end
```

See LotsaTriangles.m