

Structures

A structure is a collection of one or more variables grouped under a single name. The variables in a structure can be of different types or can themselves be structures.

Defining and Creating Structures

Structures are defined using the **struct** keyword.

```
struct dset {  
    string array sdata;  
    string array sdatadesc;  
    matrix ndata;  
    string array ndatadesc;  
};
```

```
struct company {  
    scalar id;  
    string name;
```

```
    string contact;  
    string phone;  
    string fax;  
    struct dset data;  
};
```

These statements create structure definitions that persist until the workspace is cleared. They do not create structures, only structure type definitions. To create a variable that is a structure, use:

```
struct company c;
```

This will create a variable **c** which is a structure of type *company*.

Initializing Structures

Currently, no compile time structure initialization is supported. To initialize the structure above, the following statements could be used:

```
c.id = 49;  
c.name = "XYZ Corp.";  
c.contact = "John Doe";  
c.phone = "(206) 555-1212";  
c.fax = "(206) 555-1313";  
c.data.sdata = sa;  
c.data.sdatadesc = sadesc;  
c.data.ndata = x;  
c.data.ndatadesc = xdesc;
```

The assumption is that **x** is an existing matrix and **sa**, **sadesc** and **xdesc** are existing string arrays.

Passing Structures to Procedures

Structures or members of structures can be passed to procedures. When a structure is passed as an argument to a procedure, it is passed by value. The structure becomes a local copy of the structure that was passed. The data in the structure is not duplicated unless the local copy of the structure has a new value assigned to one of its members.

Structure arguments must be declared in the procedure definition.

```
struct rectangle {
    matrix ulx;
    matrix uly;
    matrix lrx;
    matrix lry;
};

proc area(struct rectangle rect);
    retp( (rect.lrx - rect.ulx) .* (rect.uly -
rect.lry) );
endp;
```

Local structures are defined using a **struct** statement inside the procedure definition.

```
proc center(struct rectangle rect);
    struct rectangle cent;

    cent.lrx = (rect.lrx - rect.ulx) ./ 2;
    cent.ulx = -cent.lrx;
    cent.uly = (rect.uly - rect.lry) ./ 2;
    cent.lry = -cent.uly;

    retp(cent);
endp;
```
