Lecture 03.04   Exploring C—structures

C *structures* are used to group information that belongs together. The **structures** quintessential example is the *tuple*: coordinates that define a point.[2] The **tuple** following example shows some of the syntax.

```c
#include <stdio.h>

int main() {
    struct point { // declare point
        double x;
        double y;
    };
    struct point pt1 = {1.2,4.5}; // declare instance
    struct point pt2; // another instance
    pt2.x = 2*pt1.x; // assign to second instance x
    pt2.y = 3*pt1.y; // assign to second instance y
    printf("pt2 = {%f,%f}",pt2.x,pt2.y);
}
```

```
pt2 = {2.400000,13.500000}
```

The first declaration **struct** point { ... } shows that two **double** types of *members* that are grouped into a **struct**ure with *structure* **members** *tag* point. The structure tag allows us to re-use this template for further **structure tag** **struct**ure declarations, as with pt1 and pt2—two *instances* of point. **instances** Although, in this case, the two members are of the same type (**double**), they need not be.

An instance of a **struct**ure can be assigned at declaration, as with pt1, or it can be assigned after declaration, as with pt2. The members of an instance are accessed and written-to via the name defined in the initial declaration, as in pt2.x and pt2.y.

C **struct**ures can also be nested. For instance, a line segment can be defined by two points, as shown in the following snippet, which could be interpolated into the previous main function.

```c
struct segment { // declare segment
    struct point pt1;
    struct point pt2;
} seg1;
seg1.pt1 = pt1;
```

---

[2]We follow Kernighan and Ritchie (1988, p. 129), where structures are introduced via a double (2-tuple).

```
seg1.pt2 = pt2;
printf("seg1 is from {%f,%f} to {%f,%f}",
    seg1.pt1.x,seg1.pt1.y,
    seg1.pt2.x,seg1.pt2.y
);
```

Note that we can overload the names of **struct**ure members such as pt1 and x without conflict. Furthermore, the syntax that declares seg1 can be used to declare further segments.

A function can be passed as an argument a **struct**ure, or a pointer to it, or each of its members, separately. Similarly, a function can return **struct**ures in any of these ways. Note that **struct**ure tags declared in main are available to other functions.