CSC 101 Lecture Notes Week 7

C Structures

Reading: Chapter 11

I. Intro to C Structures

A. An array has multiple elements of the *same type*.

B. A structure has multiple elements of *different types*.

C Structures, cont'd

C. Consider first example in Ch 11 of the book.

1. It's a structure for planet, with components:
   - name
   - diameter
   - number of moons
   - orbit time
   - rotation time

C Structures, cont'd

D. The example code is here:

`101/examples/structs/jupiter.c`

II. Arrays of Structures

A. A `struct` type can be the same as any other C type, such as `int` or `double` or `char *`.

B. Hence, arrays of `struct` types are just fine.

C. For example:

Arrays of Structures, cont'd

```
#define MAX_PLANETS 100

typedef struct {
    double diameter;
    Planet planets[MAX_PLANETS];
    char galaxy[STRSIZ];
} SolarSystem;
```

Defining Types in Header Files

A. The upgrade from `jupiter.c` to `our-solar-system.c` was awkward.

B. Both programs use type `Planet`.

C. Entire def had to be copied into both files.

D. We'd like to have programs *share* definitions.

Defining Types in Header Files, cont'd

E. Solution -- use `.h` files.

1. They allow definition sharing.

2. Also support clean design of larger programs.

Defining Types in Header Files, cont'd

F. Design of Planetary Program:

- `planet.h`
- `planet.c`
- `planet-test.c`

- `solar-system.h`
- `solar-system.c`
- `solar-system-test.c`