Structures

Grouping different data types

Structures

- The main purpose of structures is to relate data of different types e.g. associate an age (which is an integer) with a name (which is a string)
- Think of a music database, you want title, artist, length, composer, rating all grouped together. I would be very awkward to use 5 separate arrays

Syntax

```
typedef struct {
   char title[50];
   char artist[50];
   char composer[50];
   float length;
   int rating;
} Song;
```

This doesn't allocate memory, but instead defines a new data type that can be used like int, float etc.

Using the data type

- **`Song myFavouriteSong;` // actually create the variable**
- To access the members of the structure we use . notation.

```
myFavouriteSong.length = 2.05;
myFavouriteSong.rating = 5;
strcpy(myFavouriteSong.title, "Yesterday");
strcpy(myFavouriteSong.artist, "Beatles");
strcpy(myFavouriteSong.composer, "Sir Paul");
```

Two Steps

1. Declare a structure.
   ```
   typedef struct {  ..... } Song;
   ```
2. Declare variables that use that structure.
   ```
   Song song1, song2, song3;
   ```
3. Now use song1, song2, song3 as variables we can modify.

Array of structures

- Often when we use structures we want them in an array.
  - eg. `Song myMusic[5000];`
- Now I can store information about my whole music collection.
- We access members as following
  - `myMusic[0].length = 3.45`

Initialization of array

```
int i;

for (i=0; i<5000; i++) {
   strcpy(myMusic[i].title, "");
   strcpy(myMusic[i].artist, "");
   strcpy(myMusic[i].composer, "");
   myMusic[i].length = 0.0;
   myMusic[i].rating = 0;
};
```

Nested Structures (FYI)

- One structure can be nested inside another.

```
typedef struct {
   char name[50];
   int grade;
   Song favMusic[100];
} StudentMusic;

StudentMusic ics3uMusic[30];

strcpy(ics3uMusic[0].favMusic[0].artist, “Green Day”);
```