Lecture notes: January 20, 2017

Topics:

1. Data types
2. Strucutres

Data types

What is a data type? It is a way of indicating *what a variable is*.

When you declare a variable and give it a type:

```
int x;
```

You are saying several things:
1. *What is the set of values this variable can take on?* An `int` in C can take on integer values from -2^{31} to $(2^{31} - 1)$
2. *How much space does this variable take up?* An `int` in C occupies 32 bits (indeed, there is a relationship (for integer types) between the answer to this question and the answer to question 1.
3. *How should operations on this variable be handled?* The interpretation of various arithmetic operations can change depending on the type of the variable. Performing division on `int`s is different than performing division on `float`s:
   ```
   3 / 2 = 1 //integer division
   3.0 / 2.0 = 1.5 //floating point division
   ```

Data types also help programmers understand what their code is doing.

Structures

C does not have many built in datatypes: `int`, `char`, `short`, `long`, `float`, `double`, and a few others, as well as arrays of each and pointers to each. But what if you want to talk about more complex pieces of data?

What if we want to represent a point on a graph? We cannot represent that point with just a single value, like a `float`. We need *two* values: an x coordinate and a y coordinate:

```
float point_x;
float point_y;
```

But what *is* a point? It's not a `float`. It's a `float` representing its x coordinate *and* a `float` representing its y coordinate. Can we define data types that let us say that a variable is <thing one> *and* <thing two> *and* <thing three>?

C *structures* let us do this. We can define a *new type* that lets us say that if a variable is a point, it is two floats!

```
typedef struct {
  float x;
  float y;
} Point;
```

And now when we declare a new variable, we can say that it *is a Point*:

```
Point p1;
Point p2;
```

To access the components of a structure, we use '.':

```
p1.x = 2.5;
p1.y = 3.7;

p2.x = p1.x - 3;
p2.y = p1.x * 2;
```