Using Pointers to Send Information Back From a Function

Discussion: The usual way for a function to send information back to its caller is by using a ***return*** statement.

For example, the code below on the left calls the function ***foo*** (which is shown on the right) and captures the returned value in a variable ***x***:

```
double x;

x = foo(...);
```

```
double foo(...) {
    ...
    return ...;
}
```

Notice the key features: a ***return*** statement in the function (and hence a ***non-void return type*** in the function prototype), and an ***assignment*** of the returned value into a variable in the calling code. (Note: sometimes you can just use the returned value in an expression, without bothering to store it in a variable.)

Another, less common, way for a function to send information back to its caller is by using pointers. This second way is useful in either of the following situations:

1. You want to send more than one piece of information back from the function.
2. The information being sent back is part of a large thing, e.g. an array or a structure.

See Using Pointers To Save Time and Space for more about the latter. We'll focus on:

Situation: You want to send more than one piece of information back from the function.

Here's how to do this:

- The caller has a ***variable*** of the right type to contain the information. For example:

```
float r;
```

- The caller passes the ***address*** of that variable to the function. For example:

```
foo(..., &r, ...);
```

- The function has a ***pointer*** of the right type as its corresponding parameter. For example:

```
void foo(..., float* p, ...) {
    ...
}
```

- The function sets the pointer's ***pointee*** (which is the variable in the caller) as desired. For example:

```
*p = ...;
```

See the Example for a continuation of this discussion.