Function pointers in C

940

Let's start with a basic function which we will be *pointing to*:

```
int addInt(int n, int m) {
    return n+m;
}
```

First thing, let's define a pointer to a function which receives 2 `int`s and returns an `int`:

```
int (*functionPtr)(int,int);
```

Now we can safely point to our function:

```
functionPtr = &addInt;
```

Now that we have a pointer to the function, let's use it:

```
int sum = (*functionPtr)(2, 3); // sum == 5
```

Passing the pointer to another function is basically the same:

```
int add2to3(int (*functionPtr)(int, int)) {
    return (*functionPtr)(2, 3);
}
```

We can use function pointers in return values as well (try to keep up, it gets messy):

```
// this is a function called functionFactory which receives parameter n
// and returns a pointer to another function which receives two ints
// and it returns another int
int (*functionFactory(int n))(int, int) {
    printf("Got parameter %d", n);
    int (*functionPtr)(int,int) = &addInt;
    return functionPtr;
}
```

But it's much nicer to use a `typedef`:

```
typedef int (*myFuncDef)(int, int);
// note that the typedef name is indeed myFuncDef

myFuncDef functionFactory(int n) {
    printf("Got parameter %d", n);
    myFuncDef functionPtr = &addInt;
    return functionPtr;
}
```