Data Structures with C++ : CS189

Lecture 6-1: Recursion

Summary

- A recursive function is one that calls itself
  - Usually starts with a "driver"
  - Ends with a "base case"
- Driver: Usually public; does the one-time setup before recursion starts
- Base case: The condition that prevents the function from calling itself anymore
  - "Infinite recursion" (base case failure) is even worse than an infinite loop. Every recursive call takes a little bit more memory. So eventually you run out of memory and crash ("stack overflow").

Counter - examples

- Recursion is not meant to replace loops
  - Every loop *could* be written recursively
  - Recursive is objectively worse if you don't *need* it
- Factorials (p171)
  - x! = x * (x - 1)!
  - for(int i = 0; i < x; i++)
- Exponents (p174)
  - n^m = n * n^(m-1);
  - for(int i = 0; i < m; i++)
- Counting pencils
  - "1 pencil, plus count the remaining pencils"
  - "Count each pencil once"

```
int RecursiveDriver()
{
        return Recursive(1,3);
}
int Recursive(int x, int y)
{
        // Base case
        if( x >= y )
                return 0;

        // Tail recursion
        return 1 + Recursive( x+1, y );
}
```

- This is "tail recursion". The very last thing done is call itself to go again.
- When you get to the end of a normal loop, the next thing you do is go again anyway
- Tail recursion can almost always be replaced with a loop

Call Stack

```
RecurseDriver()
        Recurse(1, 3)
                Recurse(2,3)
                        Recurse(3,3)
                        return 0
                return 1 + 0
        return 1 + 1
return 2

// Recursive subtraction!
```

- Each time you recurse, you are in a DIFFERENT COPY of the function
  - This is the one thing that gets people hung up.
  - Look at the call stack, you can see each level
- Drawing this out can help you understand what is happening
  - Like I said before, drawing these abstract structures is so helpful
- Each level has no knowledge of any other level
  - And only the driver knows where to start

Secret to Recursion

- One simple step will save you hours of pain
  - Say it out loud
- If you can express the recursive function in English, you'll find that the code ends up five lines or less
  - The concept is hard. The syntax is easy

Reverse a String (p178)

"To print a string backwards, print everything but the first, then print the first."

1.	Cat: Print "at" then C	
2.	at: Print "t" then a	
3.	t: Base case, print t	t
4.	Back to 2, print a	ta
5.	Back to 1, print C	taC

Types of Recursion

- The last thing to consider after you can say your plan out loud is decide when to recurse
- Last (Tail): Do stuff, Call self
- Middle: Do some stuff, call self, do rest
- First: Call self, do stuff
- When to call yourself is entirely dependent on what you are writing
  - None are "more correct"

End

The last step is to read these slides. (Tail recursion as a loop.)