RECURSION

Calling a function again before it ends

Delroy A. Brinkerhoff

DIRECT RECURSION

- A function calls itself

```
void f()
{
    . . .
    f();
    . . .
}
```

INDIRECT RECURSION

- A chain of function calls that results in a function being called before the first call returns

```
void a()
{
    . . .
   b();
    . . .
}
```

```
void b()
{
    . . .
    c();
    . . .
}
```

```
void c()
{
    . . .
    d();
    . . .
}
```

```
void d()
{
    . . .
    a();
    . . .
}
```

REQUIREMENTS FOR RECURSION

- One or more paths through the function where recursion takes place
- One or more paths through the function where recursion *does not* take place. These are the *base cases*
  - may be implicit for simple functions
  - easy to calculate (e.g., a constant value)
- A value, typically an argument, that changes from one function call to the next

RECURSION EXAMPLE: THE FACTORIAL FUNCTION

0! = 1 (base case)

n! = 1 * 2 * 3 * . . . * (n - 1) * n

$$f(n) = \begin{cases} 1, n = 0 \ (base\ case) \\ n(n-1), n > 0 \end{cases}$$

8! = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8

THE C++ FACTORIAL FUNCTION

```
int fact(int n)
{
    if (n > 0)
        return n * fact(n - 1); // recursion
    else
        return 1;               // non-recursion
                                // (base case)
}
```

HOW RECURSION WORKS