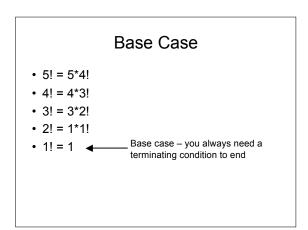
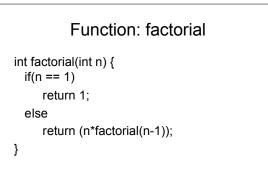


Recursion

- Idea: Some problems can be broken down into smaller versions of the same problem
- Example: n!
 - 1*2*3*...*(n-1)*n
 - n*factorial of (n-1)





Iterative Factorial

```
int factorial(int n) {
    int i;
    int product = 1;
    for(i = n; i > 1; i--) {
        product = product * i;
    }
    return product;
}
```



• Why use iteration over recursion or vice versa?

Linear Recursion

- At most 1 recursive call at each iteration
 Example: Factorial
- · General algorithm
 - Test for base cases
 - Recurse
 - Make 1 recursive call
 - Should make progress toward the base case
- Tail recursion
 - Recursive call is last operation
 - Can be easily converted to iterative

Higher-Order Recursion

- Algorithm makes more than one recursive call
- · Binary recursion
 - Halve the problem and make two recursive calls
 - Example?
- Multiple recursion
 - Algorithm makes many recursive calls
 - Example?

Rules of Recursion

- 1. Base cases. You must always have some bases cases, which can be solved without recursion.
- Making progress. For the cases that are to be solved recursively, the recursive call must always be to a case that makes progress toward a base case.
- 3. Design rule. Assume that all the recursive calls work.
- Compound interest rule. Never duplicate work by solving the same instance of a problem in separate recursive calls.

Exercises

1. Implement and test a method that uses tail recursion to convert an array of integers into their absolute values.

Towers of Hanoi

- · Three pegs and a set of disks
- · Goal: move all disks from peg 1 to peg 3
- Rules:
 - move 1 disk at a time
 - a larger disk cannot be placed on top of a smaller disk
 - all disks must be on some peg except the disk in-transit

Examples

- Design a binary recursive method for finding an element X in a sorted array A.
- Design a recursive method for printing all permutations of a given string.

 Design a recursive program to produce the following output: 0 1 1 2 1 2 3 1 2 3 4 1 2 3 1 2 3 1 2 3 1 2 3 0 1 2 3 0 1 2 3 0 1 2 0 0 1 2 	Exercises		
01 012 0123 01234 0123 012 012	1.		
0 1 2 0 1 2 3 0 1 2 3 4 0 1 2 3 0 1 2 0 1	0		
0 1 2 3 0 1 2 3 4 0 1 2 3 0 1 2 0 1	01		
0 1 2 3 4 0 1 2 3 0 1 2 0 1 2	01	2	
0 1 2 3 0 1 2 0 1	01	23	
012 01	01	234	
0 1	01	23	
	01	2	
0	01		
	0		