

Programming with Recursion



The Recursion Pattern

- ❑ **Recursion**: when a method calls itself
- ❑ Classic example: the **factorial** function:

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$$

- ❑ Recursive definition:

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot f(n-1) & \text{else} \end{cases}$$

- ❑ As a C++ method:

// recursive factorial function

```
int recursiveFactorial(int n) {  
    if (n == 0) return 1; // basis case  
    else return n * recursiveFactorial(n-1); // recursive case  
}
```

Content of a Recursive Method

❑ Base case(s)

- Values of the input variables for which we perform no recursive calls are called **base cases** (there should be at least one base case).
- Every possible chain of recursive calls **must** eventually reach a base case.

❑ Recursive calls

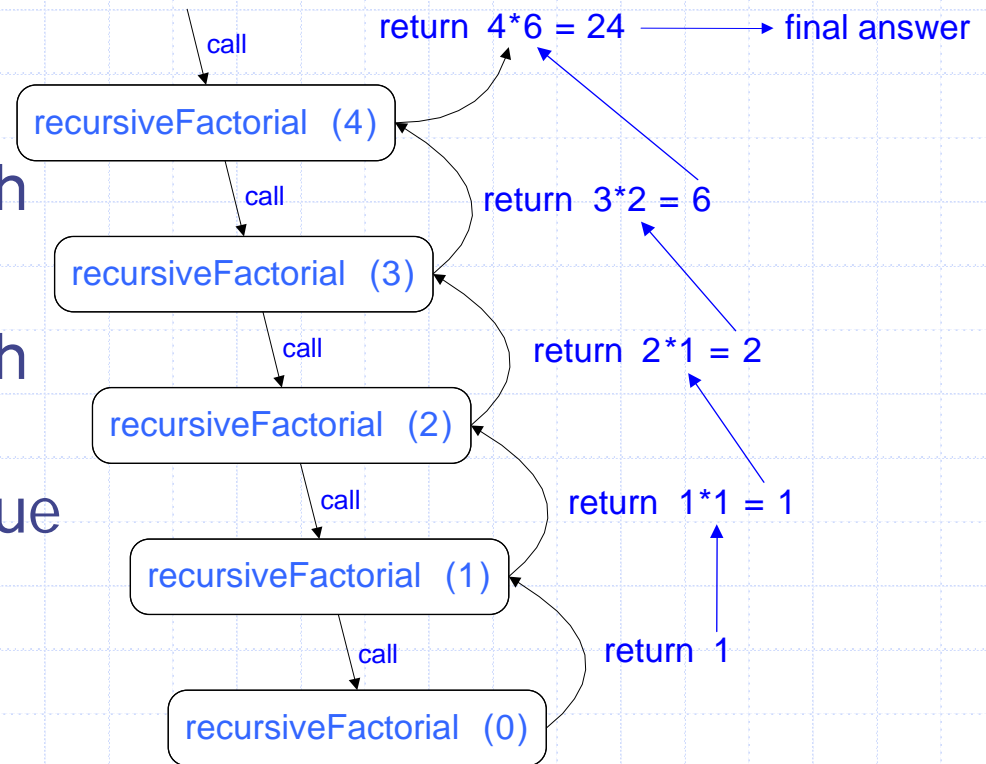
- Calls to the current method.
- Each recursive call should be defined so that it makes progress towards a base case.

Visualizing Recursion

Recursion trace

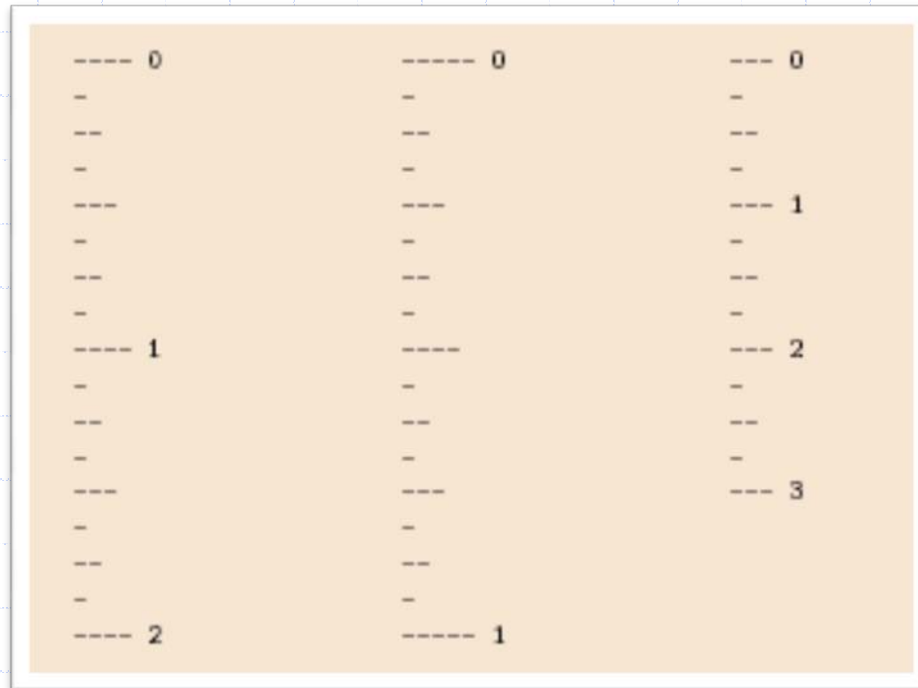
- A box for each recursive call
- An arrow from each caller to callee
- An arrow from each callee to caller showing return value

Example



Example: English Ruler

- Print the ticks and numbers like an English ruler:

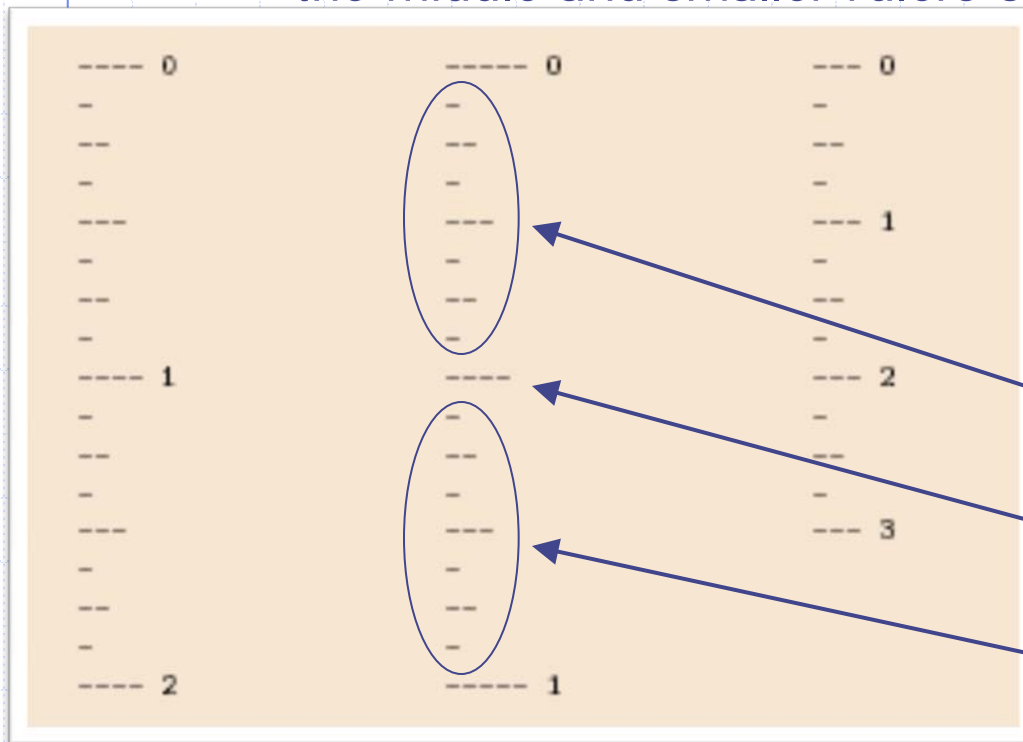


Using Recursion

`drawTicks(length)`

Input: length of a 'tick'

Output: ruler with tick of the given length in the middle and smaller rulers on either side



`drawTicks(length)`

if(length > 0) then

`drawTicks(length - 1)`

draw tick of the given length

`drawTicks(length - 1)`



C++ Implementation (1)

// draw ruler

```
void drawRuler(int nInches, int majorLength) {  
    drawOneTick(majorLength, 0);           // draw tick 0 and its label  
    for (int i = 1; i <= nInches; i++){  
        drawTicks(majorLength- 1);         // draw ticks for this inch  
        drawOneTick(majorLength, i);       // draw tick i and its label  
    }  
}
```

// draw ticks of given length

```
void drawTicks(int tickLength) {  
    if (tickLength > 0) {  
        drawTicks(tickLength- 1);          // stop when length drops to 0  
        drawOneTick(tickLength);           // recursively draw left ticks  
        drawTicks(tickLength- 1);          // draw center tick  
    }                                       // recursively draw right ticks  
}
```


C++ Implementation (2)

// draw a tick with no label

```
void drawOneTick(int tickLength) {  
    drawOneTick(tickLength, - 1);  
}
```

// draw one tick

```
void drawOneTick(int tickLength, int tickLabel) {  
    for (int i = 0; i < tickLength; i++)  
        cout << "-";  
    if (tickLabel >= 0) cout << " " << tickLabel;  
    cout << "\n";  
}
```