ECE 2400 Computer Systems Programming Spring 2025

Topic 2: C Recursion

School of Electrical and Computer Engineering
Cornell University

revision: 2025-01-29-12-07

zyBooks The zyBooks logo is used to indicate additional readings and coding labs included in the course zyBook which will not be discussed in detail in lecture. Students are responsible for all material covered in lecture and in the course zyBook.

Copyright © 2025 Anne Bracy. All rights reserved. This handout was prepared by Prof. Anne Bracy at Cornell University for ECE 2400 / ENGRD 2140 Computer Systems Programming (derived from previous handouts prepared and copyrighted by Prof. Christopher Batten). Download and use of this handout is permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial or non-commercial means requires written permission.

1. Single Recursion

Recall from mathematics, the factorial of a number (n!) is:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n > 0 \end{cases}$$

So in other words:

$0!$	$=$		$=$	1
$1!$	$=$		$=$	1
$2!$	$=$	1×2	$=$	2
$3!$	$=$	$1 \times 2 \times 3$	$=$	6
$4!$	$=$	$1 \times 2 \times 3 \times 4$	$=$	24
$5!$	$=$	$1 \times 2 \times 3 \times 4 \times 5$	$=$	120

We can write a function to calculate factorial using a `for` loop:

```
1  int factorial( int n ) {
2    int result = 1;
3    for ( int i = 1; i <= n; i++ )
4      result = result * i;
5    return result;
6  }
```

- The loop implementation does not really resemble the original mathematical formulation
- The mathematical formulation is inherently recursive
- Can we implement factorial more directly using recursion?

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n > 0 \end{cases}$$

We can use the exact same "by-hand" execution approach we learned in the previous topic to understand recursion.

```
01 int factorial( int n )
02 {
03   // base case
04   if ( n == 0 ) {
05     return 1;
06   }
07   // recursive case
08   if ( n > 0 ) {
09     return n *
10       factorial(n-1);
11   }
12 }
13
14 int main()
15 {
16   int a = factorial(3);
17   return 0;
18 }
```

stack

Questions:

- What if n is negative?
- What if the execution arrow reaches end of a non-void function without encountering a return statement?

2. Multiple Recursion

Recall from mathematics, the Fibonacci sequence is a sequence of integers such that every number after the first two is the sum of the two preceding ones:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

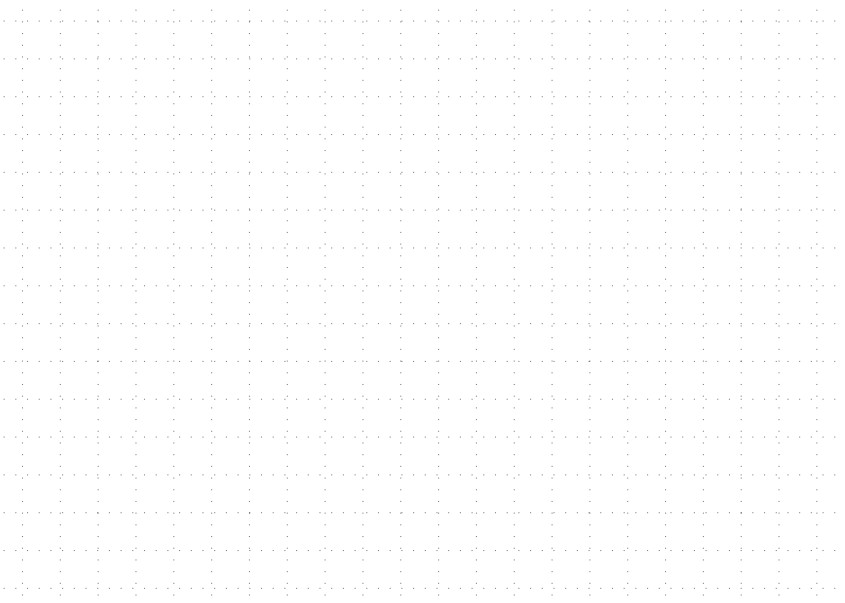
The numbers in the Fibonacci sequence are called "Fibonacci numbers". By definition, the first two numbers in the Fibonacci sequence are 0 and 1. Ancient scholars realized the importance of this sequence in both mathematics and nature. Fibonacci sequences can be found in the arrangement of leaves on a stem or patterns in a pine cone.

We can write a function to calculate the n^{th} Fibonacci number using a `for` loop:

```
1  int fib( int n ) {
2
3    // by definition
4    if (n == 0) return 0;
5    if (n == 1) return 1;
6
7    int fib_minus2 = 0;
8    int fib_minus1 = 1;
9    int result     = 0;
10
11   for ( int i=2; i<=n; i++ ) {
12
13     result = fib_minus1
14            + fib_minus2;
15
16     fib_minus2 = fib_minus1;
17     fib_minus1 = result;
18
19   }
20   return result;
21 }
```

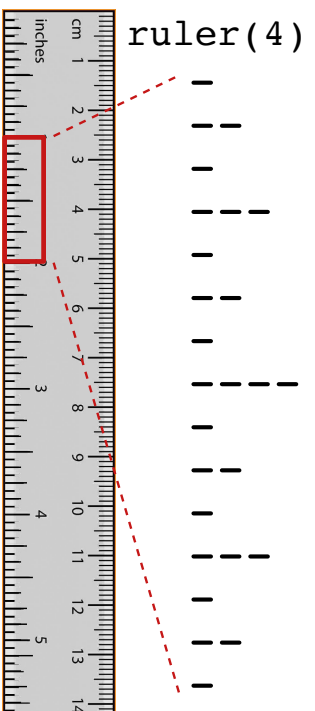
Can we implement `fib` more elegantly using recursion?

Illustrating call tree for `fib`



3. Writing a Recursive Function

Write pseudo-code for a recursive function which draws the tick marks on a vertical ruler. The middle tick mark should be the longest and mark the 1/2 way point, slightly shorter tick marks should mark the 1/4 way points, even slightly shorter tick marks should mark the 1/8 way points and so on. The function should take one argument: the height of the middle tick mark (i.e., the number of dashes). The function should always return 0.

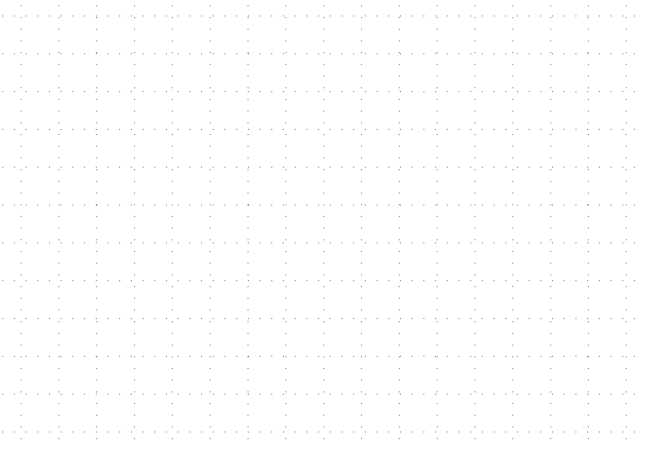



```
int ruler( int height ) {
```

```
ruler(1)    ruler(2)    ruler(3)    ruler(4)    ruler(5)
  -           -           -           -           -
              --          --          --          --
              -           -           -           -
                          ---         ---         ---
                          -           -           -
                          --          --          --
                          -           -           -
                                      ----        ----
                                      -           -
                                      --          --
                                      -           -
                                      ---         ---
                                      -           -
                                      --          --
                                      -           -
                                                  -----
                                                  -
                                                  --
                                                  -
                                                  ---
                                                  -
                                                  --
                                                  -
                                                  ----
                                                  -
                                                  --
                                                  -
                                                  ---
                                                  -
                                                  --
                                                  -
```

- Step 1: Work an example yourself
  - height = 2, height = 3
- Step 2: Write down what you just did
  - What is the base case?
  - What is the recursive case?
- Step 3: Generalize your steps
  - for any height
- Step 4: Test your algorithm
  - does it work for height = 4?
- Step 5: Translate to pseudocode

Think about the recursive call tree?



Manually work through example ruler

zyBooks The course zyBook includes a coding lab to implement a recursive algorithm to print a down/up sequence.