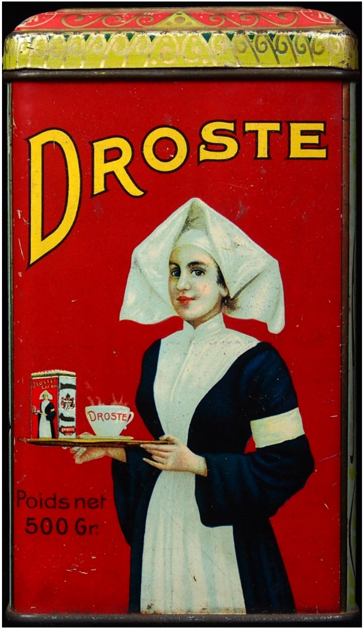
Recursion

C1

- Recursive functions & algorithms
- The call stack

- “...embedding an expression of some type within an expression of the same type” (*linguistics*)



(Alf van Beem, CC0, via Wikimedia Commons)

- The body of a function can contain function calls, including *calls to the same function*.
  - This is known as recursion.
- The function must have a branching statement, such that a recursive call does not always take place (“base case”); otherwise, recursion never ends.
- Recursion is a way to think about solving a problem: how to reduce it to a simpler instance of itself?

Example: Factorial function

- Compute

$$\begin{aligned} f(n) &= n * (n-1) * (n-2) * \ldots * 1 \\ &= n * f(n-1) \end{aligned}$$

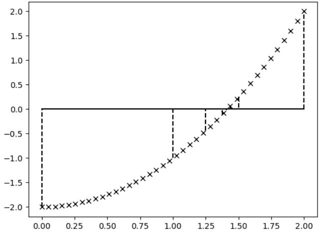
- Base case:

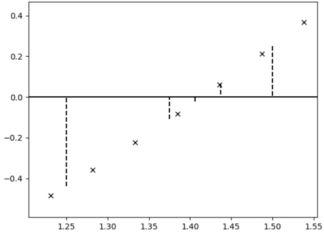
$$f(1) = 1$$

```
static int f(int n) {
  if (n == 1)
    return 1;
  else
    return n * f(n-1);
}
```

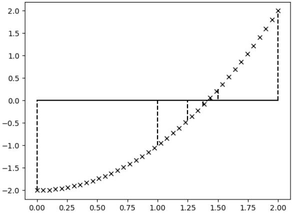
Example: Solving an equation

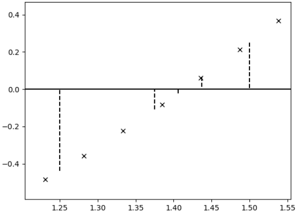
- Solve $g(x) = 0$.
- For example, find x such that $(x^2 - 2) = 0$.
- The interval-halving algorithm (a.k.a. binary search).





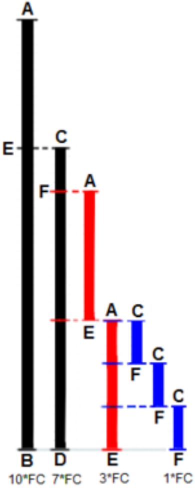
- Assumption: $g(x)$ is monotone increasing and crosses 0 in the interval $[l, u]$.
- Idea:
  - Find the middle of the interval, m:
  - if $g(m) \approx 0$, we're done;
  - if $g(m) < 0$, the solution lies in $[m, u]$;
  - if $g(m) > 0$, the solution lies in $[l, m]$.





Example: gcd

- The *greatest common divisor* (*gcd*) of numbers a and b is the greatest number c such that $a = ic$ and $b = jc$, for integers i and j.
- Euclid's algorithm.

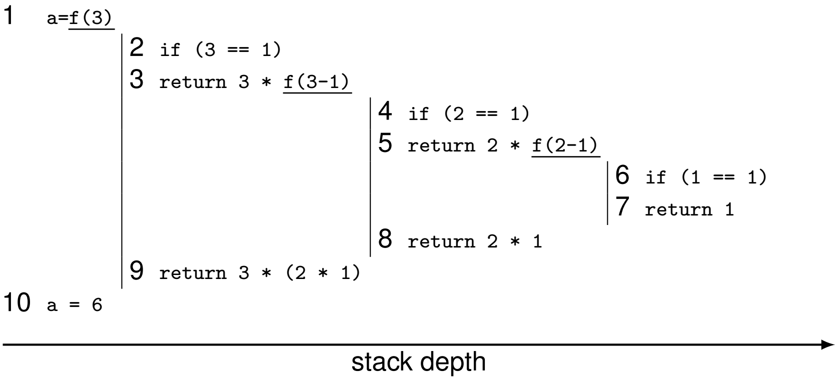


- Assumption: a and b are positive, $a < b$.
- Euclid's idea:
  - Find m and r such that $b = ma + r$, $0 \leq r < a$.
    - `m = b / a` (b/a rounded down)
    - `r = b % a` (remainder of b/a)
  - If $r = 0$, then $\mathrm{gcd}(a, b) = a$.
  - Else, $\mathrm{gcd}(a, b) = \mathrm{gdc}(r, a)$.

The call stack

- When a function call begins, the current instruction sequence is put “on hold” while the the function body executes.
- When the function ends, it returns to the next instruction after where the function was called.
- The “to-do list” of where to come back to after each current function call is called the stack.
- Variables declared in a function (including its parameters) are local to each call to that function.

```
static int f(int n) {
  if (n == 1)
    return 1;
  else
    return n * f(n-1);
}
```



```
1  a=f(3)
      2 if (3 == 1)
      3 return 3 * f(3-1)
                    4 if (2 == 1)
                    5 return 2 * f(2-1)
                                  6 if (1 == 1)
                                  7 return 1
                    8 return 2 * 1
      9 return 3 * (2 * 1)
10 a = 6
```

preview: Recursive data structures

- A recursive data structure is made up of parts that reference other parts of the same type.
- Example: A binary tree is
  - a leaf node; or
  - a node with two children, that are binary trees.