ETH Zürich
Chair of Software Engineering

Einführung in die Programmierung
Introduction to Programming

Prof. Dr. Bertrand Meyer

Exercise Session 9

Today

- Feedback on the mock exam
- Recursion
  - Recursion
    - Recursion
      - Recursion
        - Recursion

Recursion: an example

- Fibonacci numbers:

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

- How can we calculate the n-th Fibonacci number?

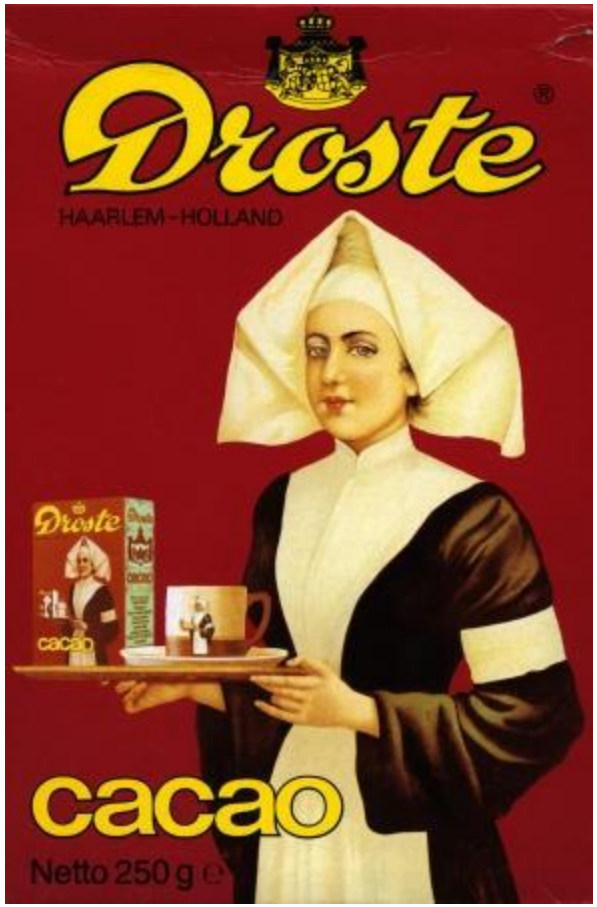
- Recursive formula:

$$F(n) = F(n-1) + F(n-2) \quad \text{for } n > 1$$

$$\text{with } F(0) = 0,\ F(1) = 1$$

Recursion: a second example

- Another example of recursion

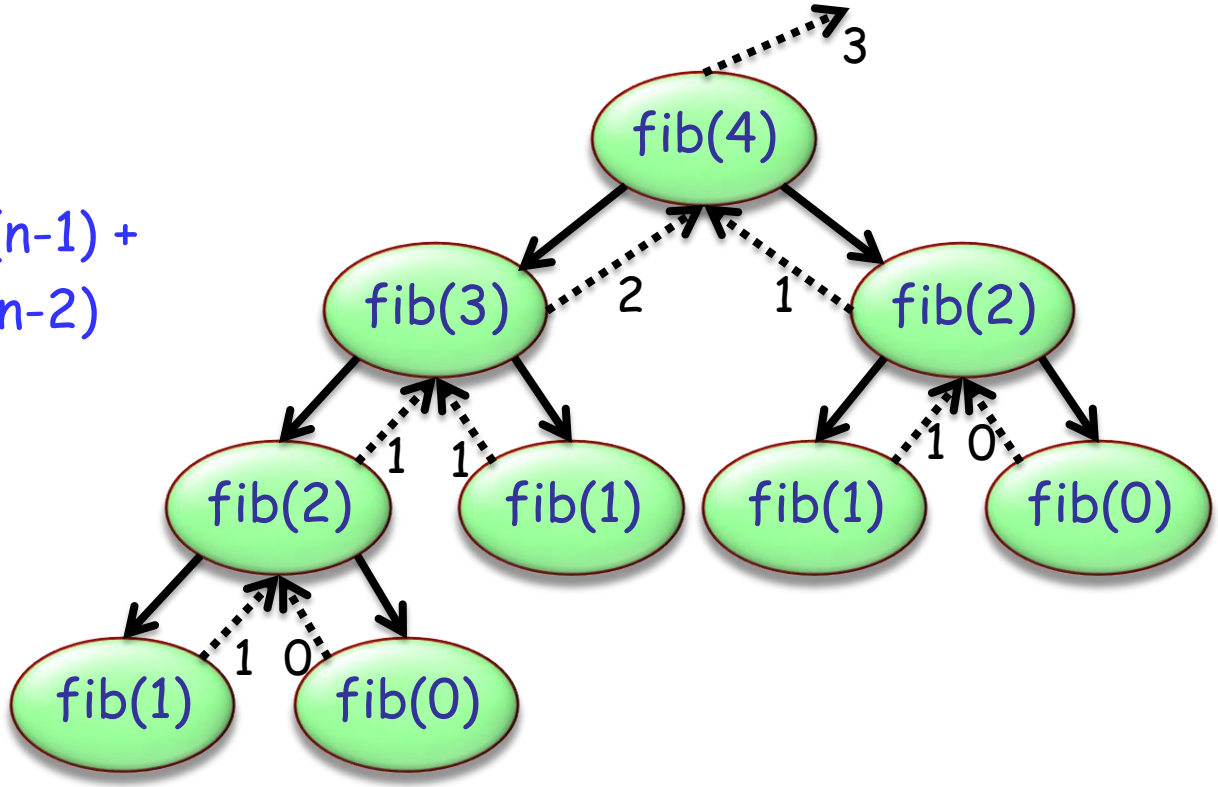


Source: en.wikipedia.org/wiki/Recursion

A recursive feature

```
fibonacci(n: INTEGER): INTEGER
  do
    if n = 0 then
      Result := 0
    elseif n = 1 then
      Result := 1
    else
      Result := fibonacci(n-1) +
                fibonacci(n-2)
    end
  end
```

- Calculate fibonacci(4)



The general notion of recursion

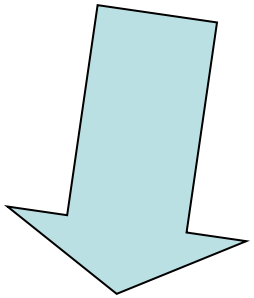
A definition for a concept is recursive
if it involves an instance of the concept itself

- The definition may use more than one "*instance of the concept itself*"
- *Recursion* is the use of a recursive definition

Thoughts

„To iterate is human, to recurse - divine!"

but … computers are built by humans 



Better use iterative approach if reasonable?

Iteration vs. recursion

- Every recursion could be rewritten as an iteration and vice versa.
- BUT, depending on how the problem is formulated, this can be difficult or might not give you a performance improvement.

Exercise: Printing numbers

- If we pass n = 4, what will be printed?

```
print_int (n: INTEGER)
    do
        print (n)
        if n > 1 then
            print_int (n - 1)
        end
    end
```

4321

```
print_int (n: INTEGER)
    do
        if n > 1 then
            print_int (n - 1)
        end
        print (n)
    end
```

1234

Exercise: Reverse string

Hands-On

- Print a given string in reverse order using a recursive function.

Exercise: Solution

```
class APPLICATION

create
    make

feature
    make
        local
            s: STRING
        do
            create s.make_from_string ("poldomangia")
            invert(s)
        end

    invert (s: STRING)
        require
            s /= Void
        do
            if not s.is_empty then
                invert (s.substring (2, s.count))
                print (s[1])
            end
        end
end
```

Exercise: Sequences

- Write a recursive and an iterative program to print the following:

  111,112,113,121,122,123,131,132,133,
  211,212,213,221,222,223,231,232,233,
  311,312,313,321,322,323,331,332,333,

- Note that the recursive solution can use loops too.

Exercise: Recursive solution

```
cells: ARRAY [INTEGER]

handle_cell (n: INTEGER)
    local
        i: INTEGER
    do
        from
            i := 1
        until
            i > 3
        loop
            cells [n] := i
            if (n < 3) then
                handle_cell (n+1)
            else
                print (cells [1].out+cells [2].out+cells [3].out+",")
            end
            i := i + 1
        end
    end
```

Exercise: Iterative solution

```
from
    i := 1
until
    i > 3
loop
    from
        j := 1
    until
        j > 3
    loop
        from
            k := 1
        until
            k > 3
        loop
            print (i.out+j.out+k.out+",")
            k := k + 1
        end
        j := j + 1
    end
    i := i + 1
end
```