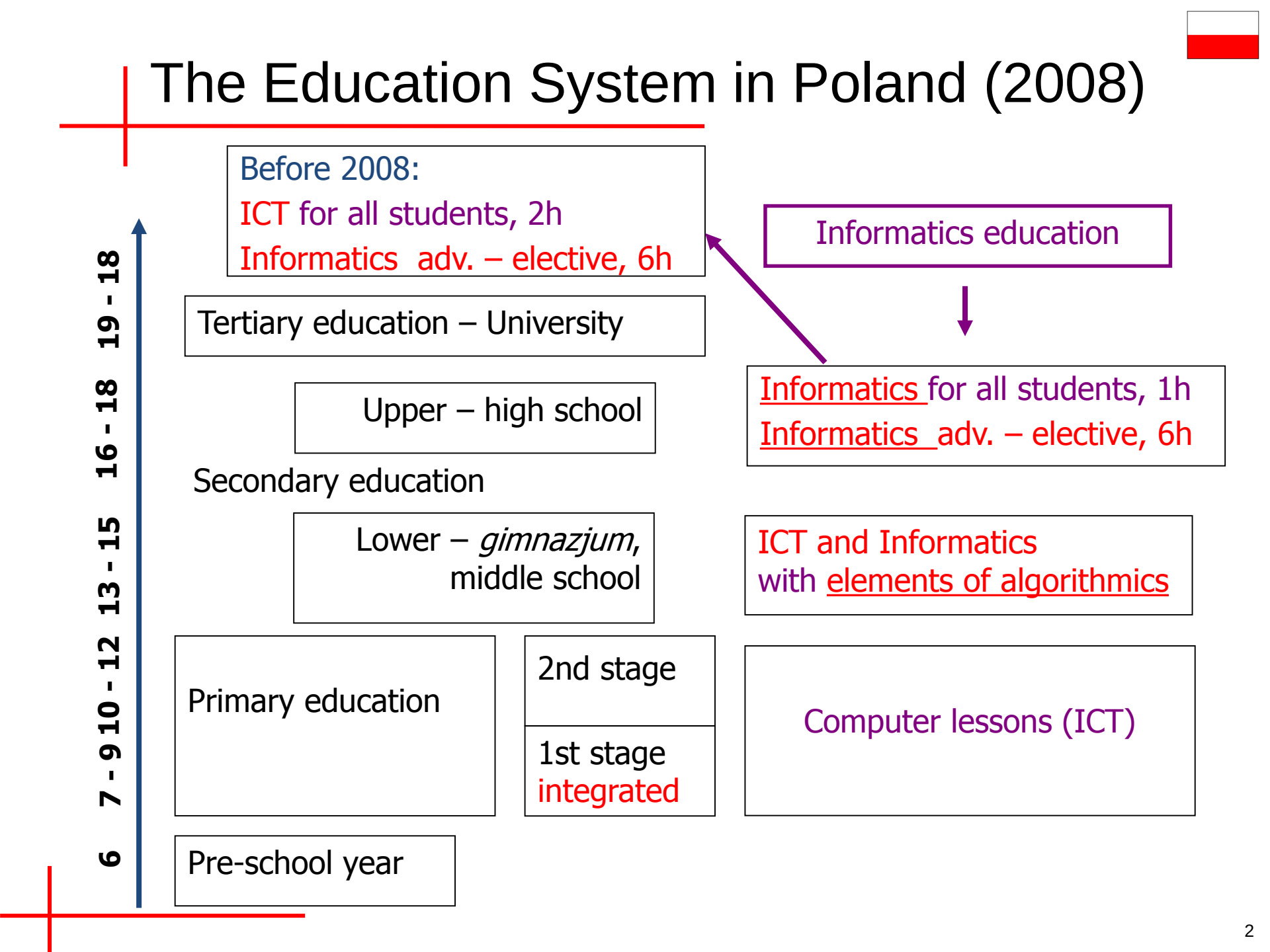
Introducing recursion

Maciej M. Sysło, Anna Beata Kwiatkowska
University of Wrocław, UMK Toruń
{aba}, {syslo}@mat.umk.pl, http://mmsyslo.pl/

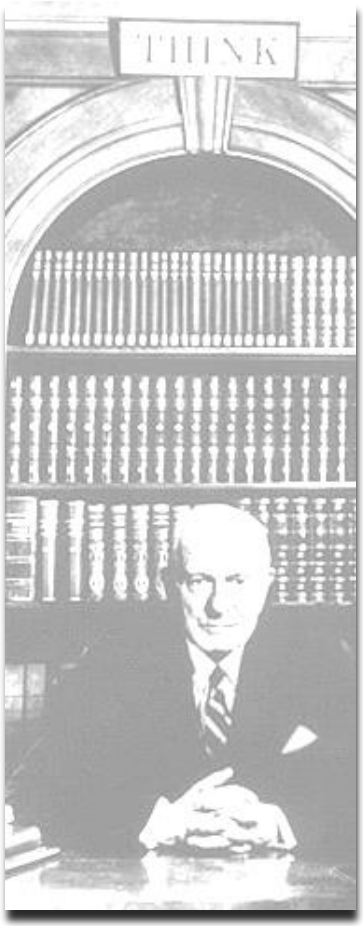
The Education System in Poland (2008)



Computational thinking (CT)

Includes a range of mental tools (J. Wing, 2006) originated in CS:

- reduction and decomposition of complex problems
- approximation, when exact solution is impossible
- recursion: inductive thinking, used in reduction
- representation and modeling
- heuristic reasoning



Recursion – CS Unplugged

Ershov, 1988:

```
eat porridge;
  if the plate is empty then STOP
  else
      eat a spoonful of porridge;
      eat porridge
```

Syslo, 2009:

```
dance;
  if the music is not played then STOP
  else
      make a step;
      dance
```

Recursion – school topics (in PL curriculum)

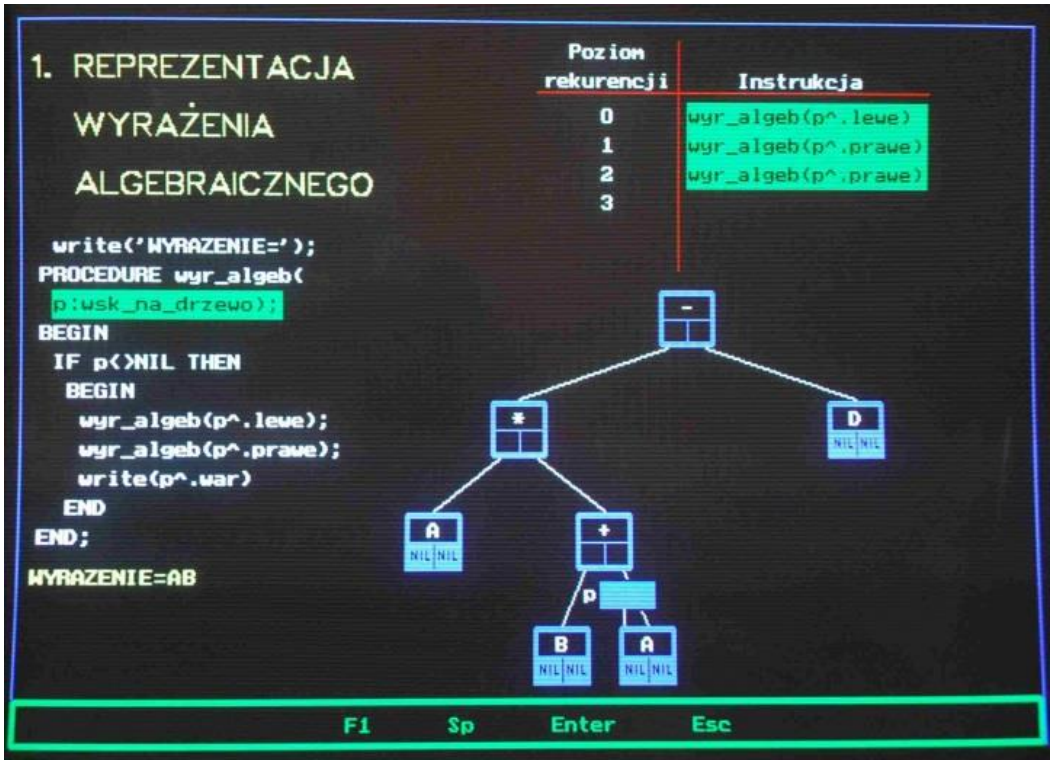
- Fibonacci numbers (+recursive definition)
- the Hanoi Tower (+recursive complexity relation)
- generating expressions in RPN
- GCD – Euclid's Algorithm
- generating permutations (+n!)
- binary search (+optimal algorithm)
- merge sort
- quick sort
- printing a number digit-by-digit
- Horner's rule
- (really) fast exponentiation

Visualization of recursion

Constructing the RPN form from an expression tree (1993)

trace

recursive procedure

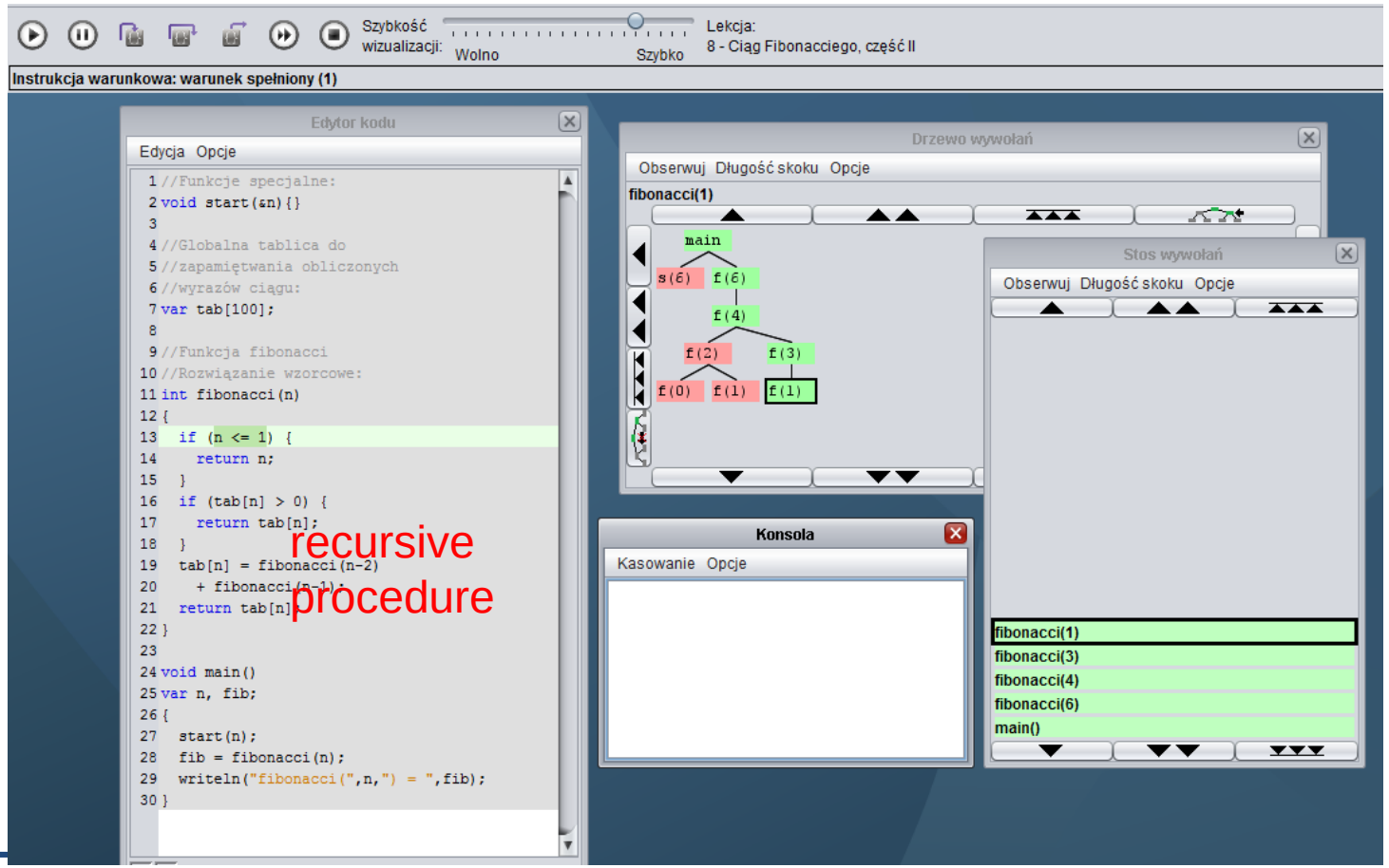


expression tree
recursion tree

Visualization of recursion

Fibonacci numbers (2013)

expression tree
recursion tree



Recursion *versus* iteration

Ershov, 1988:

```
eat porridge;
  if the plate is empty then STOP
  else
       eat a spoonful of porridge;
       eat porridge
```

Eating porridge:

```
while the plate is not empty do
  eat a spoonful of porridge
```

Syslo, 2009:

```
dance;
  if the music is not played then STOP
  else
       make a step;
       dance
```

Dance:

```
while music is played do
  make a step
```

Recursion *versus* iteration

Fibonacci numbers:

Iteration:

$F_1 = 1,\ F_2 = 1,$

$F_i = F_{i-1} + F_{i-2},$ for $i = 3, 4, 5, \ldots$

for $i = 5$: $F_1\ F_2\ F_3\ F_4\ F_5$

recursion:

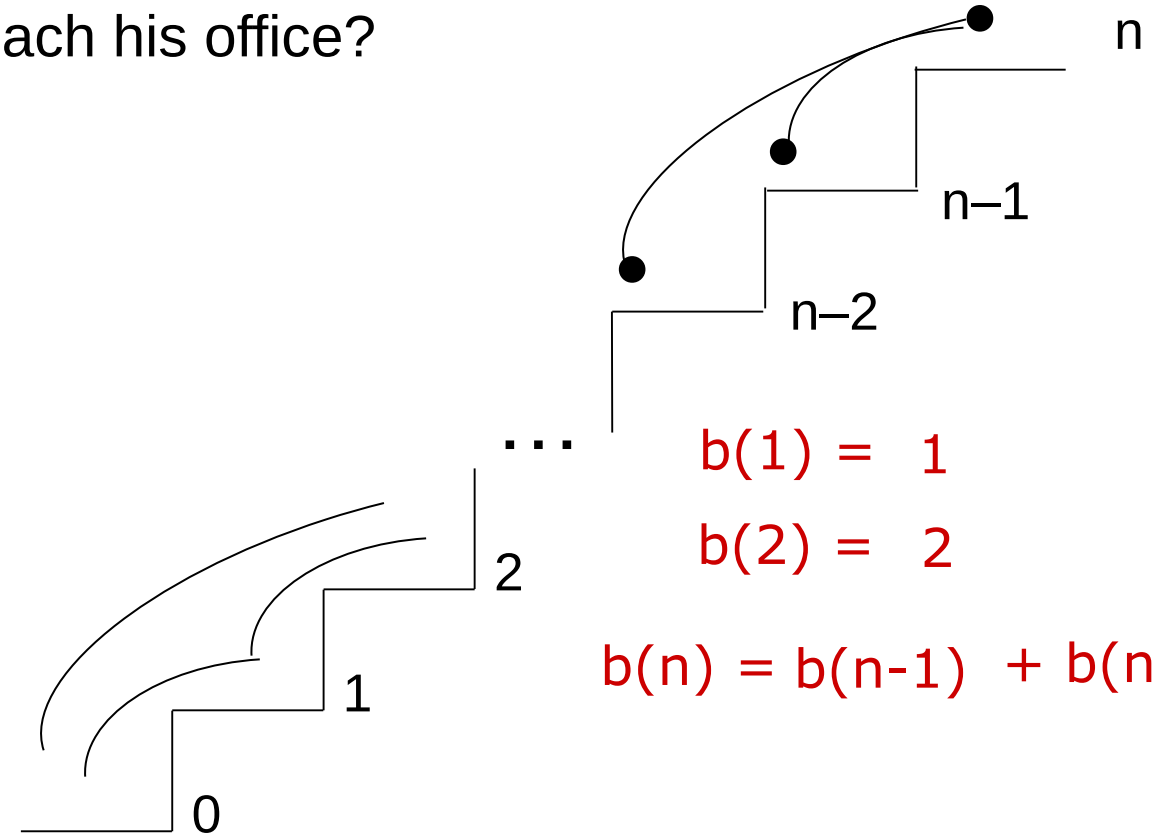
$F_1 = 1,\ F_2 = 1,$

$F_i = F_{i-1} + F_{i-2},$ for $i > 2$

for $i = 5$: $F_5\ F_4\ F_3\ F_2\ F_1\ F_2\ F_3\ F_2\ F_1$

Recursive thinking

Story: A professor S. has his office on the second floor and the staircase from the first floor to the second floor consists of 12 (in general *n*) steps. He is a very chaotic person and he takes one or two steps at a time. In how many ways can he reach his office?



$b(1) = 1$

$b(2) = 2$

$b(n) = b(n-1) + b(n-2)$ for $n > 2$

Recursive thinking – CS example

Task: print an integer, digit by digit, starting with the most important digit

```
procedure Digits(n:int);
  if n < 10 then write(n)
  else begin
    Digits(n div 10);
    write(m mod 10)
  end
```

```
procedure Digits(n,p:int);
  if n < p then write(n)
  else begin
    Digits(n div p,p);
    write(m mod p)
  end
```

Related tasks, extensions:

- print digits of a number starting from the less important one
- print digits of a decimal number in the representation with respect to the base *p*
- compute the number of ones in the binary representation
- compute the sum of digits of a number in its representation with respect to any given base *p*.

Powerful recursion

How many multiplications are needed to calculate the value of the exponential function?

In RSA (for a small exponent):

$$x^{12345678912345678912345678912345}$$

In maths class:

$$x^n = x*x*x* \ldots *x$$

n – 1 multiplications: 12345678912345678912345678912344

Supercomputer 10^{15} = 1 000 000 000 000 000 000 oper/sek

12345678912345678,912345678912344 sekund¶
12345678912345678,912345678912344/60 = 205761315205761,31520576131520567 minut¶
205761315205761,31520576131520567/60 = 3429355253429,35525342935525342 78 godzin¶
3429355253429,355253429355253427 8/24 = 142889802226,22313555955646889282 dób¶
142889802226,22313555955646889282/365 = 391478910,20883050838234649011733 lat,¶

It will take $3*10^8$ years

Powerful recursion

```
Power(x,n)                          { x^n }
   if n=1 then Power:=x
   else if n - even then
        Power:=Power (x,n/2)^2              {x^n = (x^(n/2))^2}
        else Power:=Power(x,n-1)*x          {x^n = (x^(n-1))x}
```

Number of multiplications: :

number of bits in a binary representation of n – $\log_2 n$

plus

number of 1's in a binary representation of n – $\log_2 n$

Total: at most $2*\log_2 n$ multiplications

For

$x^{12345678901234567890123456789012345}$

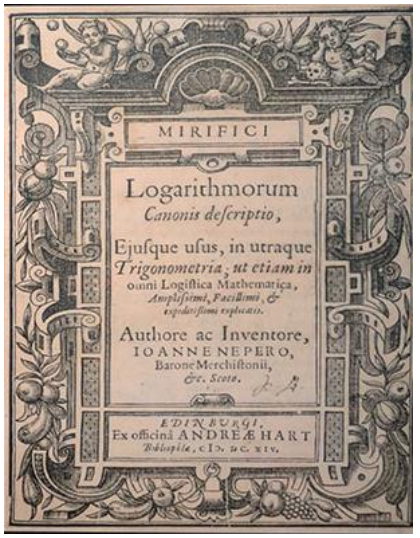
Only …. 200 multiplications – Shock!

For $n = 10^{500}$, $\log n \leq 2000$

m	$\log_2 m$
10^4	10
10^6	20
10^9	30
10^{12}	40
10^{20}	66,5
10^{50}	166
10^{100}	332,2

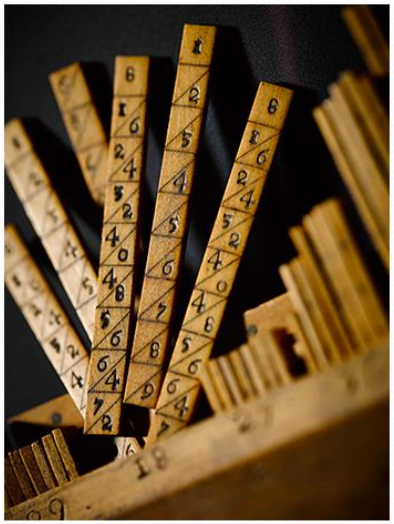
John Napier – Logarithm John and his tools

John Napier
1550-1617



His book of 1614
on Logarithmorum

400th anniversary of inventing logarithm



Napier's rods or bones are multiplication tables written out on rods divided into squares. they were used to aid multiplication and divis to work out cube and s

Napier's rods

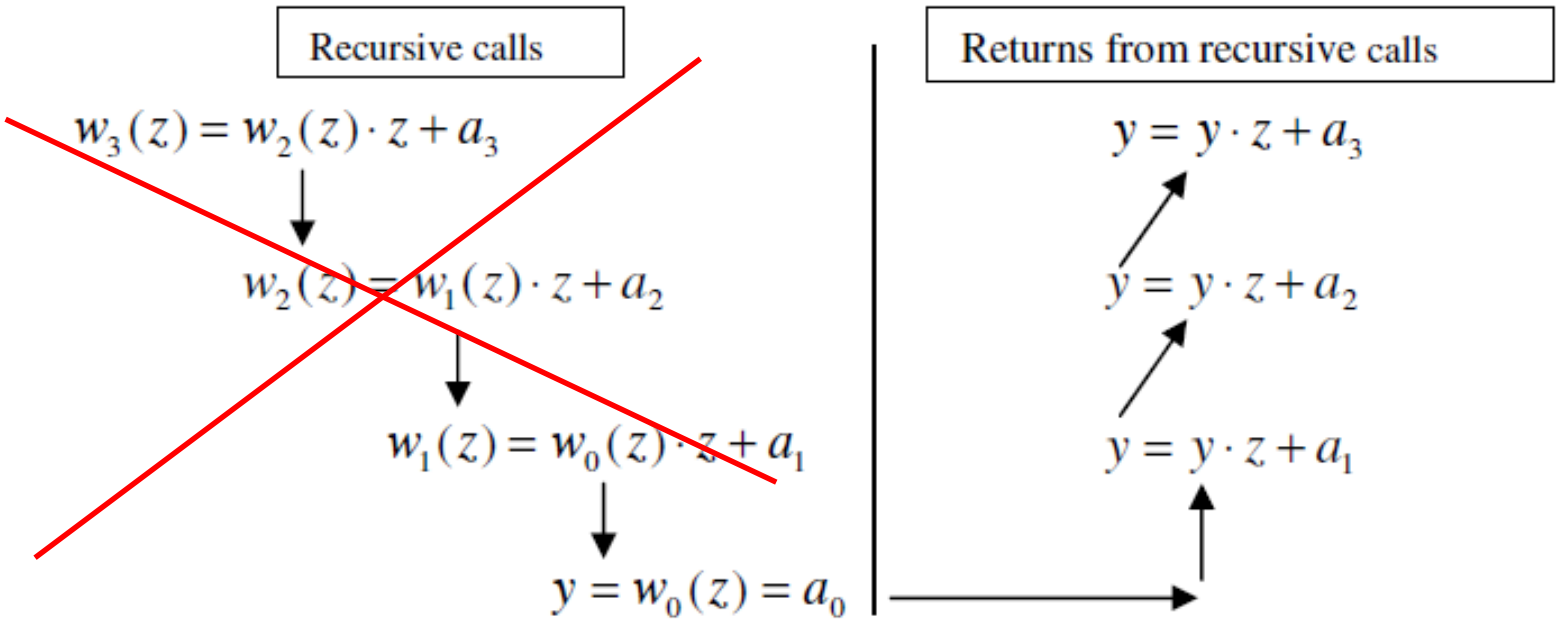
This calculating instrument is the oldest signed scientific instrument made in Scotland.

The oldest calculating instrument made in Scotland

Recursion in programming – warning

Recursive Horner's rule:

$$w_0(x) = a_0$$
$$w_n(x) = w_{n-1}(x) \cdot x + a_n \text{ for } n \geq 1.$$



Iterative Horner's rule

Recursion in programming – warning and hope

Task: Calculate F_n

Answers:

(1) Using $F_n = F_{n-1} + F_{n-2}$

- recursive – non-polynomial algorithm
- iterative – linear time algorithm

Suggestion – replace a linear time by logarithmic time

(2) Logarithmic recursion (divide-and-conquer):

$$F_0 = 0, F_1 = 1$$

$$F_{2n-1} = F_{n-1}^2 + F_n^2$$

$$F_{2n} = 2F_{n-1}F_n + F_n^2$$

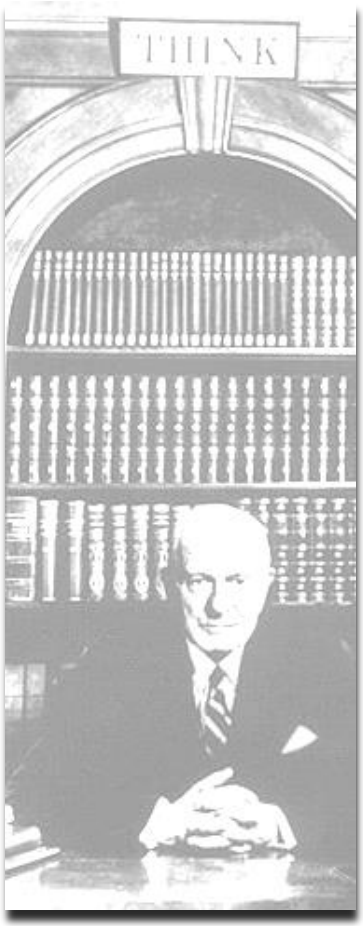
- Recursive – non-polynomial
- Iterative – logarithmic !!!

We have applied:

- algorithmic thinking
- recursive thinking
- reductive thinking
- logarithmic thinking
- design thinking
- …

… other mental tools

of computational thinking



Thank you for your attention

and don’t forget to: