Information Technology for Digital Humanities
Lecture 5

Mario Verdicchio
Università degli Studi di Bergamo
Academic Year 2023-2024

Lecture 5 (October 10 2023)

- Binary system exercises

Binary encoding of numbers

Numbers with base 10:

$$215 = 2\times10^2 + 1\times10^1 + 5\times10^0$$

Numbers with base 2:

$$110010111 = 1\times2^8 + 1\times2^7 + 0\times2^6 + 0\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$$

Exercise 1

Convert the following numbers from base 2 to base 10: 101, 1000, 11011.

$101_2 \rightarrow ?_{10}$

$$\begin{array}{ccc} 1 & 0 & 1 \\ 2 & 1 & 0 \\ \downarrow & \downarrow & \downarrow \end{array}$$

$$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$

$$4 + 0 + 1 = 5_{10}$$

$$1000_2 = ?_{10}$$

$$\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 \end{array}$$

$$1 \cdot 2^3 = 8_{10}$$

$$\underbrace{1\;1\;0\;1\;1}_{4\;\;3\;\;2\;\;1\;\;0}{}_{2} = ?_{10}$$

$$2^4 + 2^3 + 2^1 + 2^0 =$$

$$16 + 8 + 2 + 1 = 27_{10}$$

$$\underbrace{1\,1\,0\,1\,1}_{4\quad 3\quad 2\quad 1\quad 0}{}_{2} = ?_{10}$$

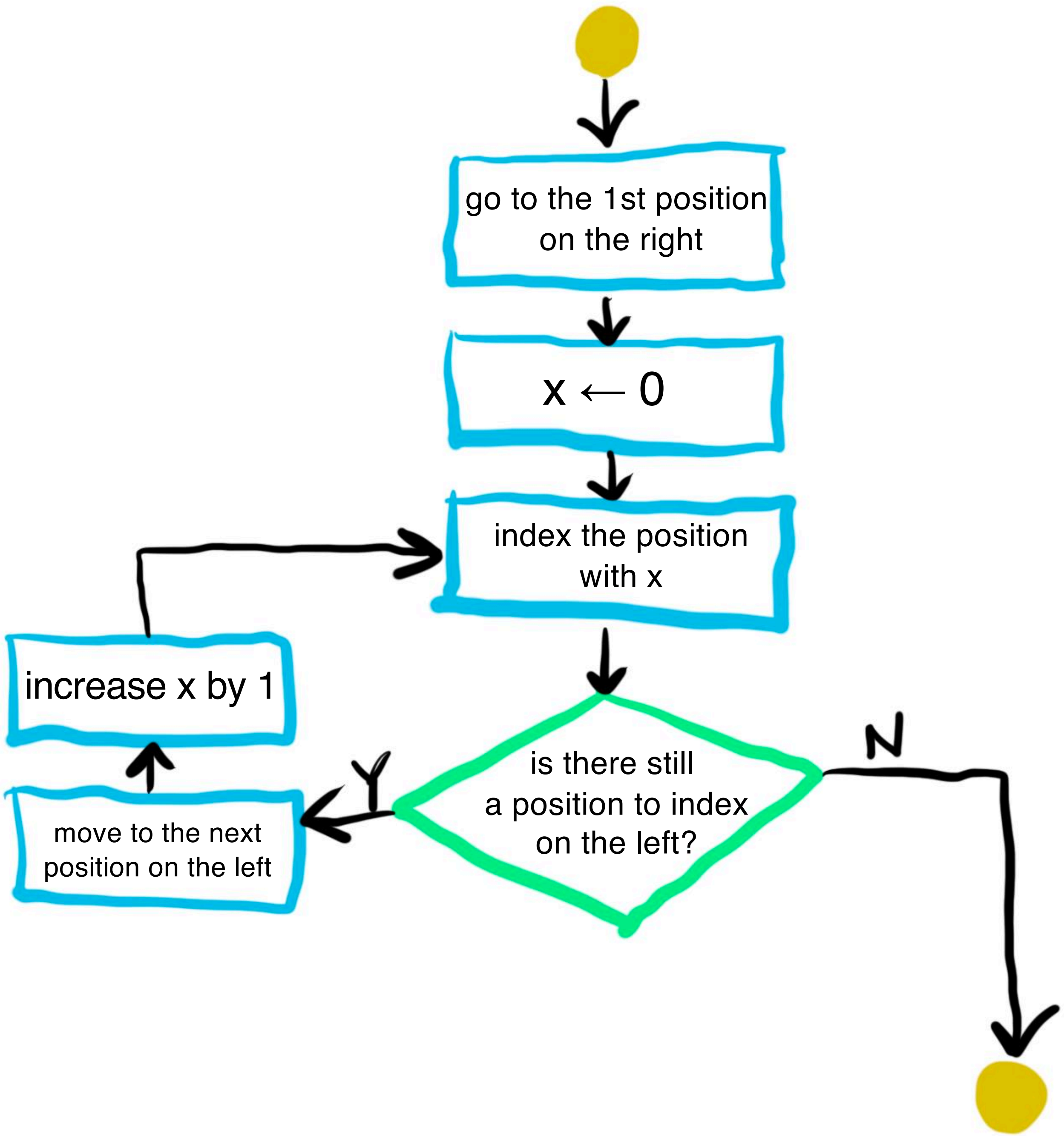
Assigning a number to the positions of the binary number (also known as “indexing”, that is, establishing an index) may look like a trivial task, but it presents significant differences whether it is done by a human or by a computer.

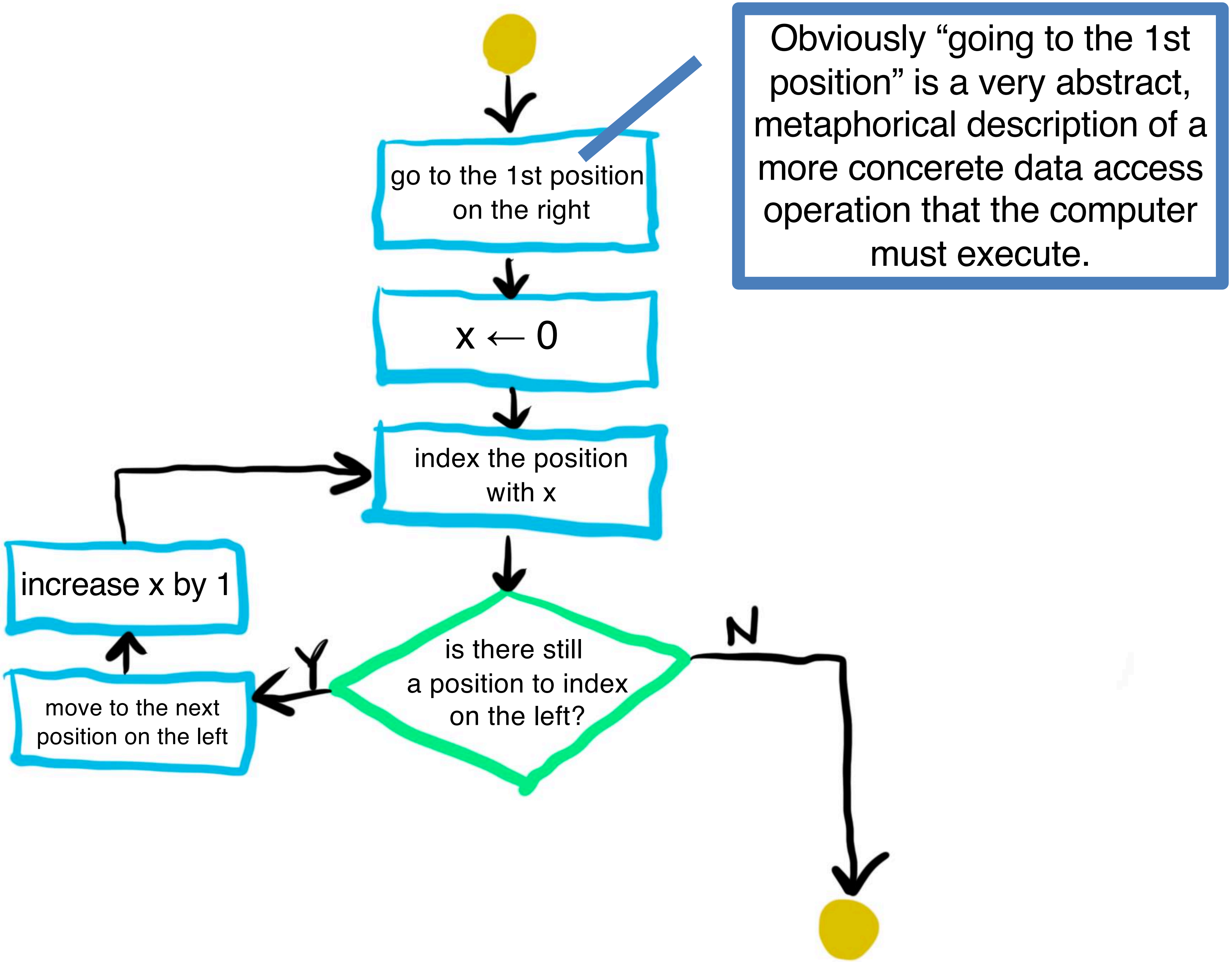
$$11011_2 = ?_{10}$$

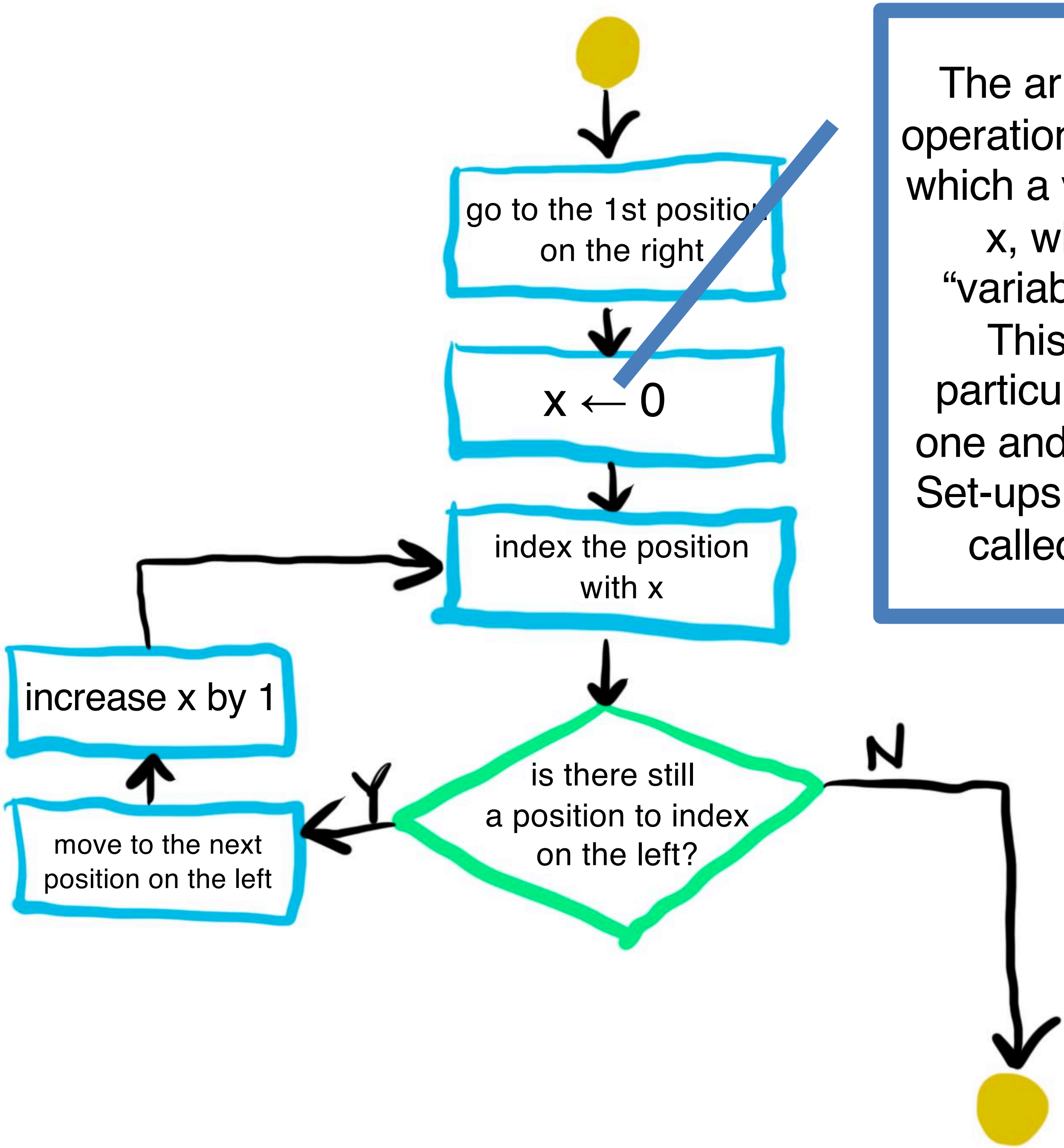
4 3 2 1 0

A human can typically use their eyes to understand how many positions are there at a glance, and then starts writing the indexes from right to left, starting with “0”. A computer cannot do that, because they do not have a global perception, but can only treat one data after the other.

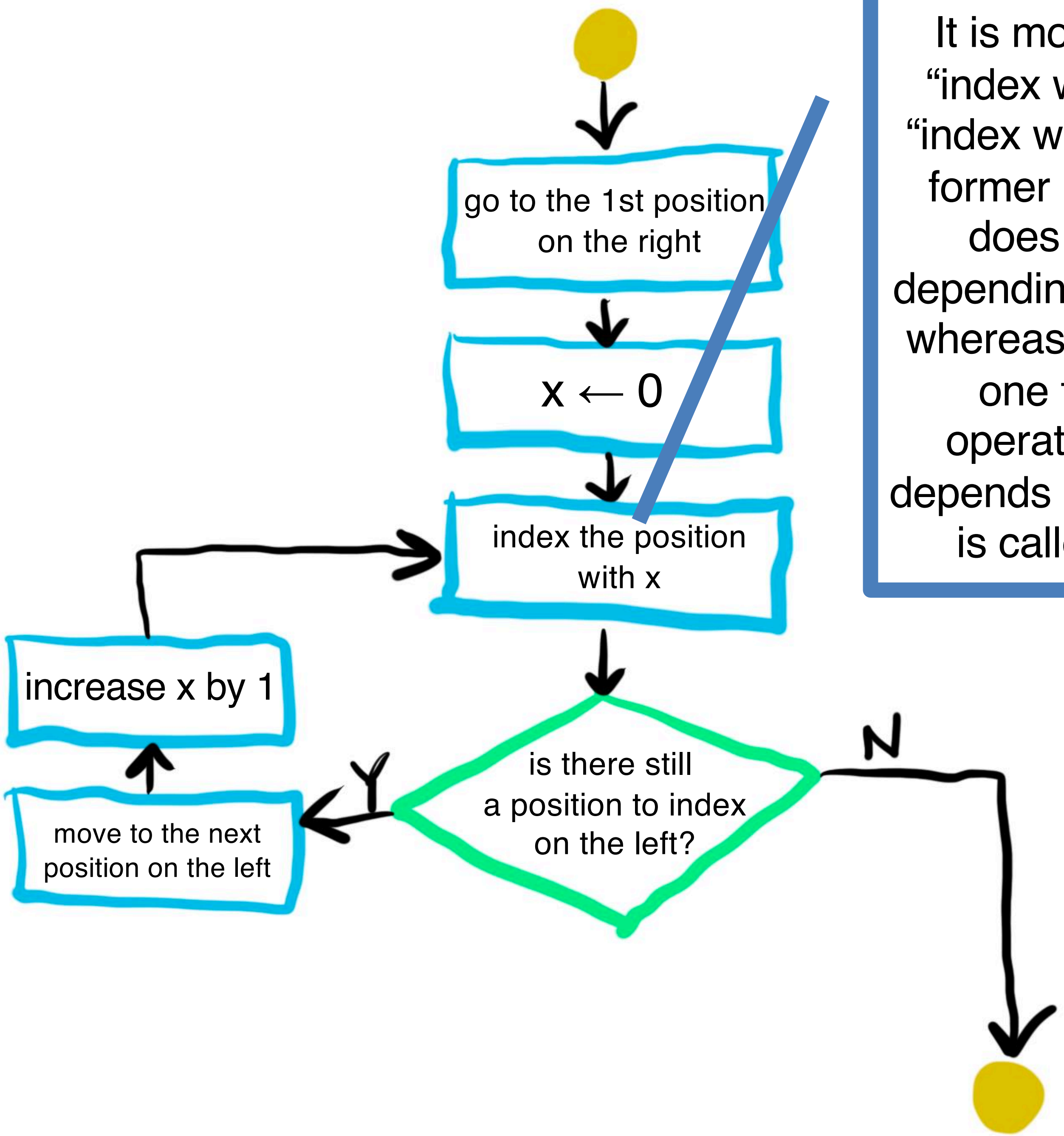
In the following slide, there is an algorithm that guides a computer in the task of indexing the positions.


go to the 1st position
on the right
x ← 0
index the position
with x
is there still
a position to index
on the left?
N
Y
move to the next
position on the left
increase x by 1

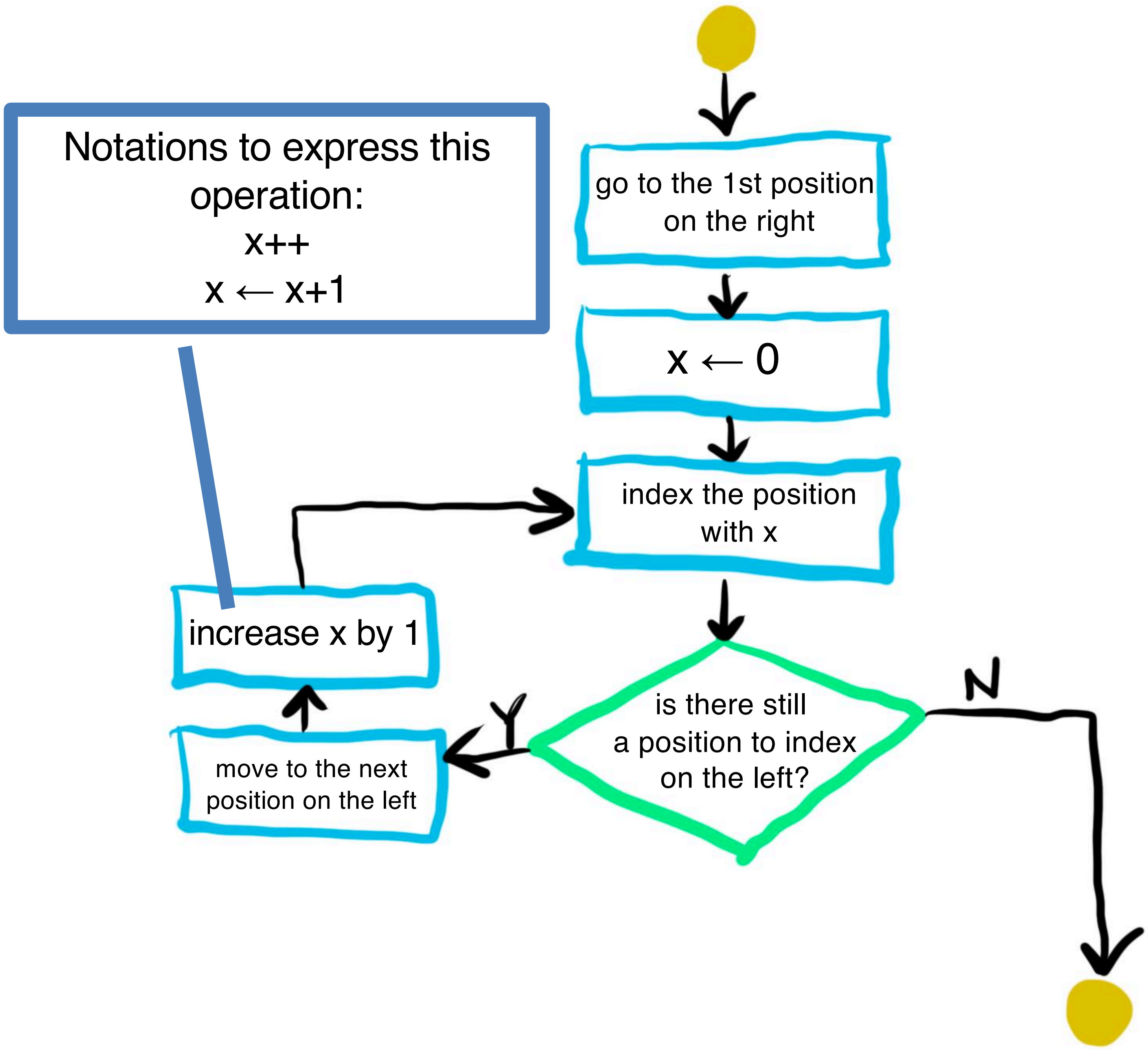

go to the 1st position on the right
x ← 0
index the position with x
is there still a position to index on the left?
Y
N
move to the next position on the left
increase x by 1
Obviously “going to the 1st position” is a very abstract, metaphorical description of a more concerete data access operation that the computer must execute.



The arrow represents an operation of “assignment”, in which a value is assigned to x, which works as a “variable” or “parameter. This assignment, in particular, is the very first one and works as a set-up. Set-ups in an algorithm are called “initializations”.



It is more clever to have “index with x” rather than “index with 0”, because the former is more flexible: it does different things depending on the value of x, whereas the latter has only one fixed result. An operation whose result depends on a parameter in it is called “parametric”.


Notations to express this operation:
x++
x ← x+1
go to the 1st position on the right
x ← 0
index the position with x
increase x by 1
is there still a position to index on the left?
Y
N
move to the next position on the left

Exercise 2

Convert the following numbers from base 10 to base 2: 8, 23, 144, 201.

While the definition of a binary system helps us solve Exercise 1, we must find new methods to solve Exercise 2. There are two.

First method:

given the number n in base 10, we look for the largest power of 2 that is less than or equal to n.

If it is the same, we have solved the problem: we write a 1 in the position corresponding to that power of 2, followed by zeros.

For example, 8 is a power of 2: 2^3 to be precise, so its binary encoding will be 1000.

However, if the largest power of 2 that is less than or equal to n is less than n (let's call it k), let's set it aside and we calculate the difference n-k.

We repeat the same procedure with n-k, and look for the largest power of 2 that is less than or equal to it.

We continue until we are able to express n as a sum of powers of 2.

We take the list of powers and write a 1 in the corresponding positions, 0 in the others.

For example, the largest power of 2 contained in 23 is 16 (2^4). Their difference is 7, in which 4 (2^2) is contained. The difference is 3 where there is 2 (2^1), after which only 1 (2^0) remains.

Writing the powers of 2 present in order we get 10111.

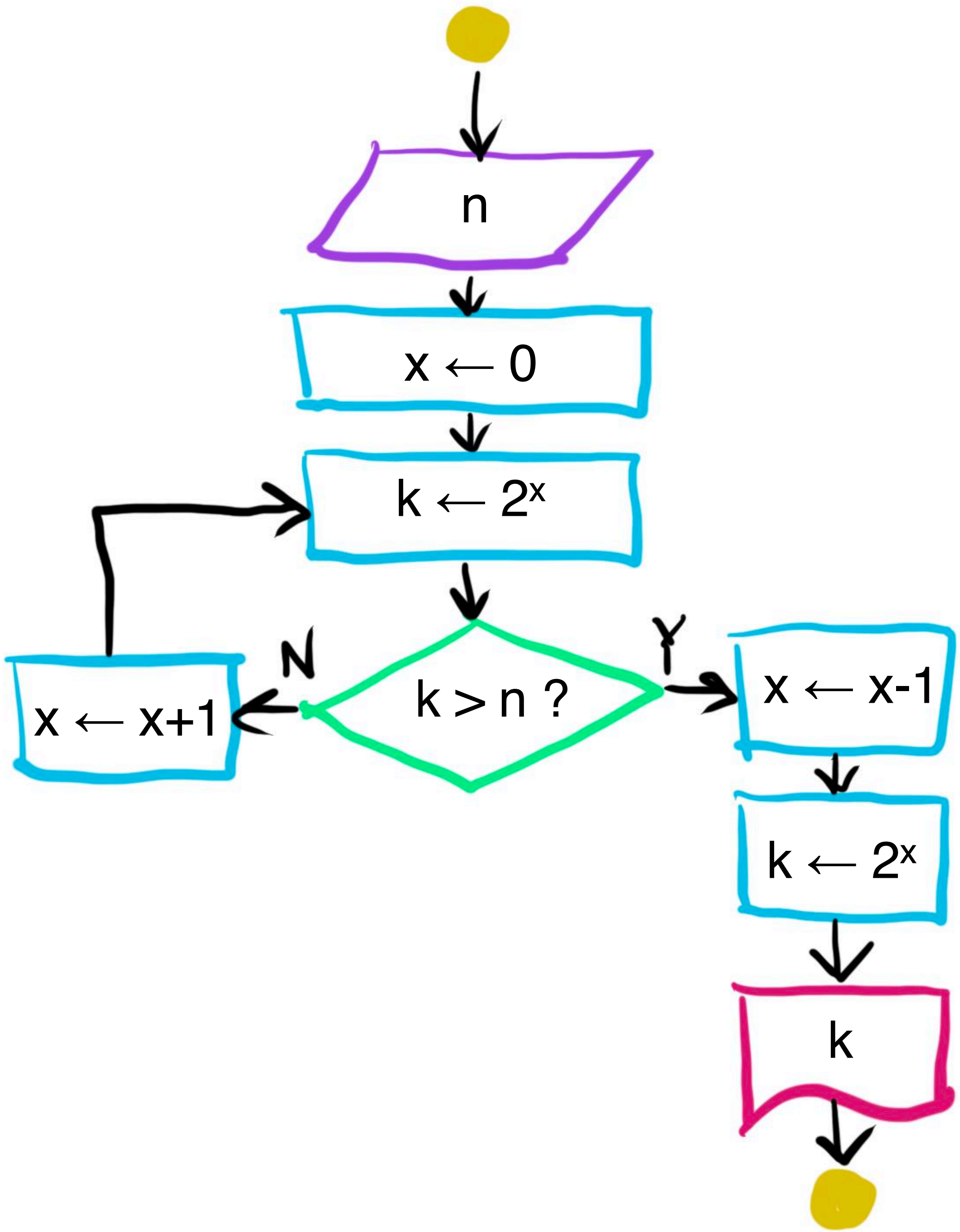
While the definition of a binary system helps us solve Exercise 1, we must find new methods to solve Exercise 2. There are two.

First method:

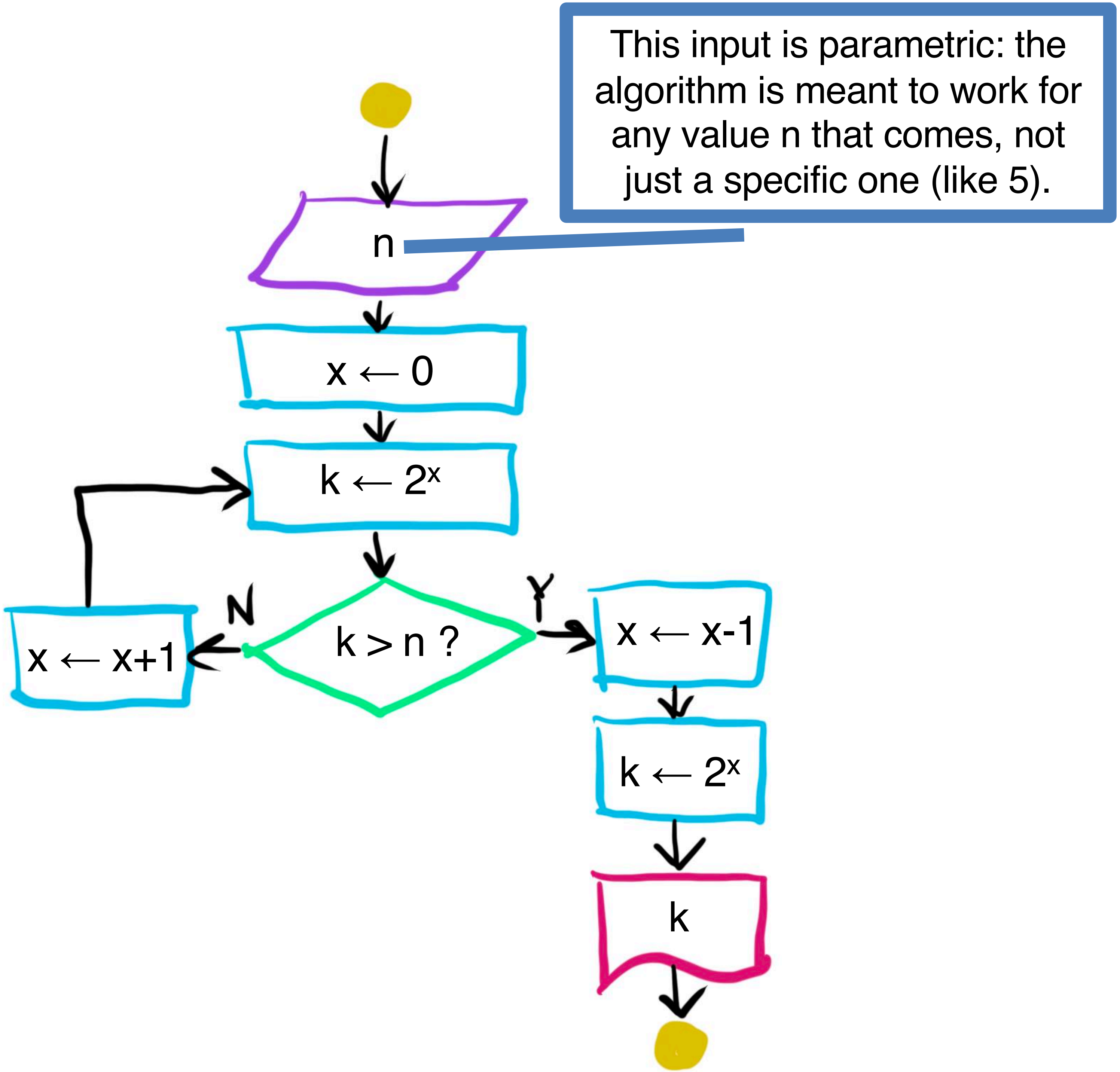
given the number n in base 10, we look for the largest power of 2 that is less than or equal to n.

If

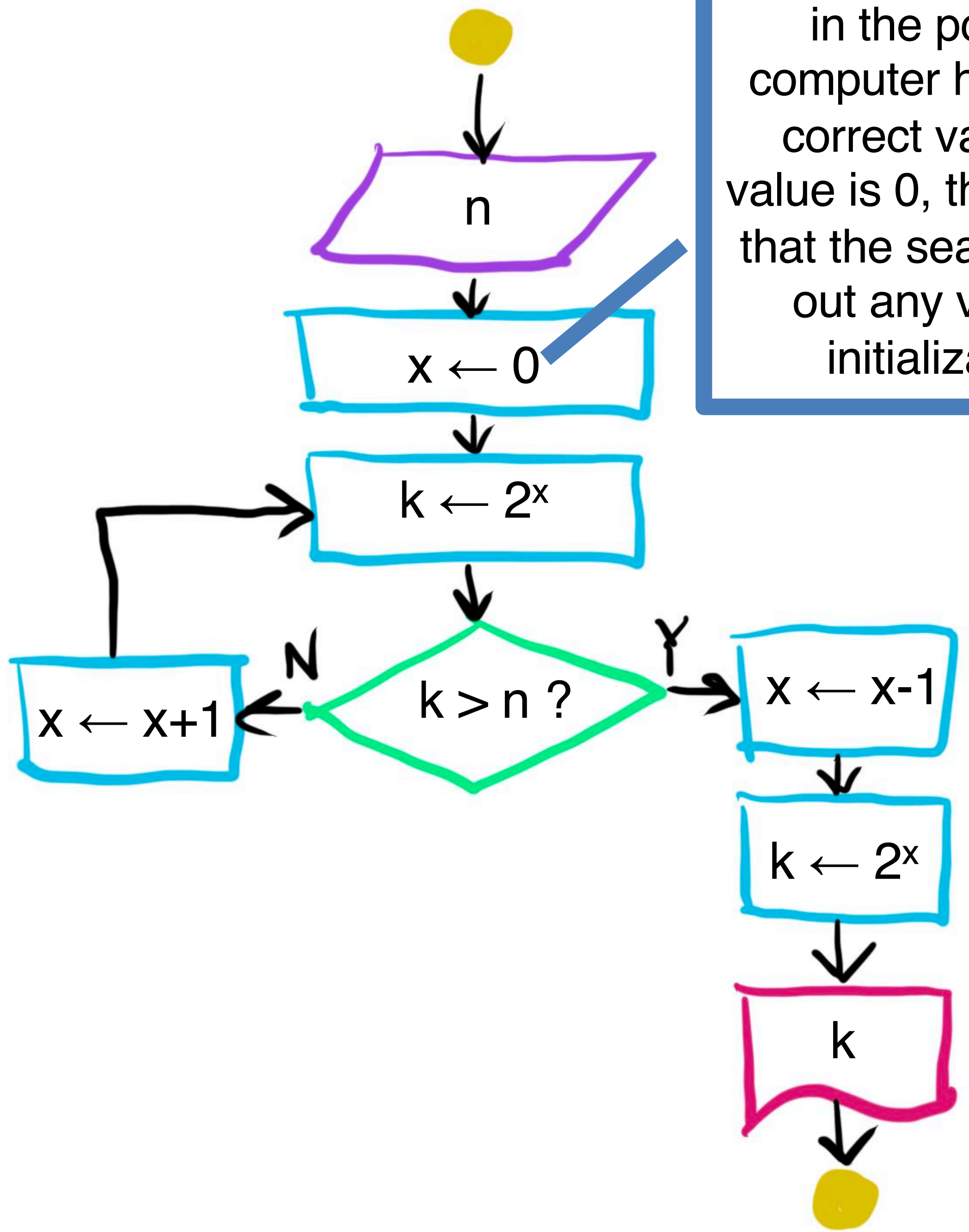
This is another task that can be executed in many different ways.
Humans that are very acquainted with arithmetic and powers of 2, they do it swiftly, almost unconsciously.
For example, if n is 5, then it comes immediately to my mind that the largest power of 2 that is less than or equal to 5 is 4, because I have been working with powers of 2 for a long time.
However, a computer, despite being a calculator, does not have experience, nor intuition, nor a consciousness, not a subconscious mind.
Hence, given n, for a computer to find the largest power of 2 that is less than or equal to n, we need to specify an algorithm like the one in the next slide.

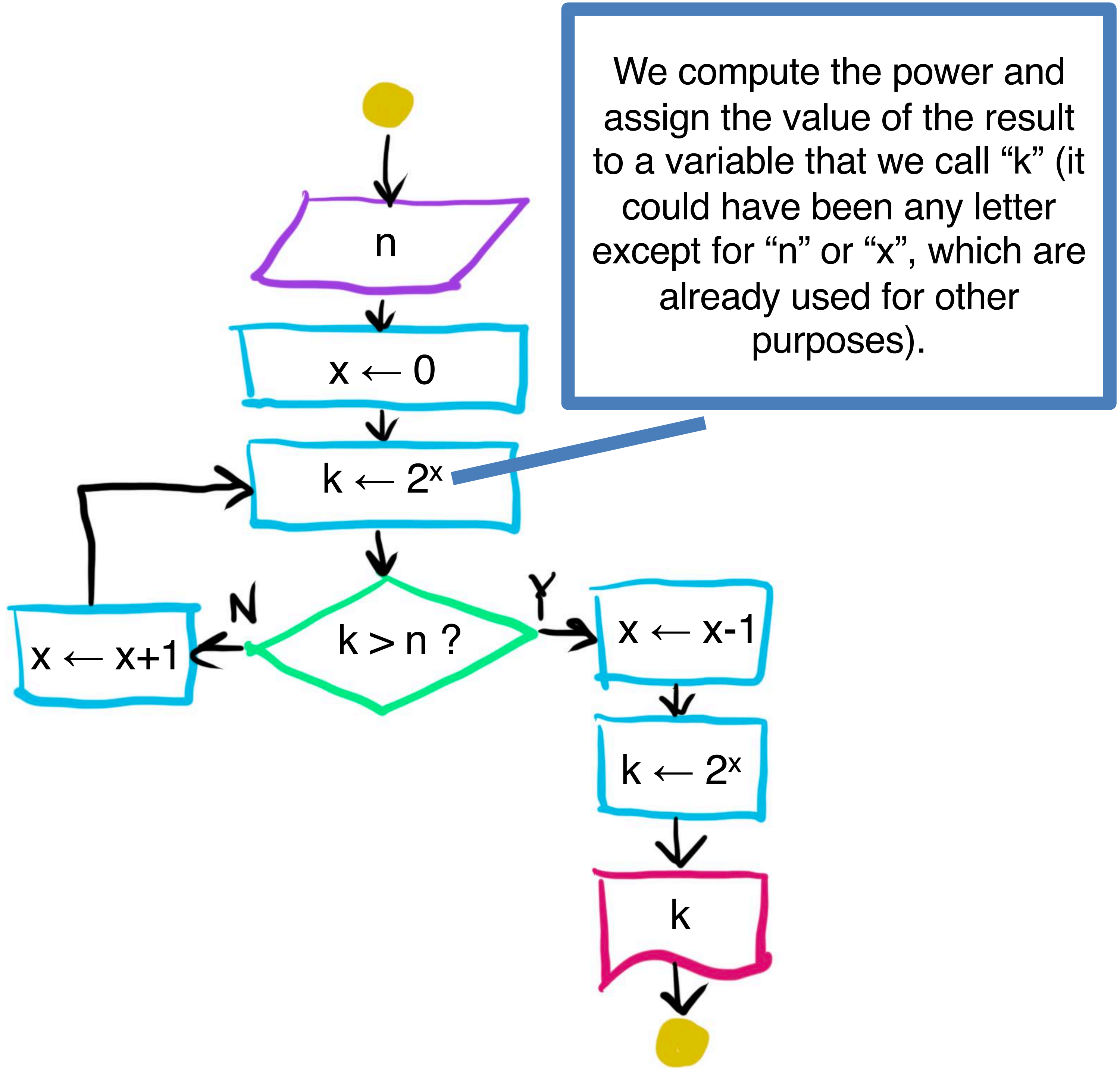


n
x ← 0
k ← 2^x
N
k > n ?
Y
x ← x+1
x ← x-1
k ← 2^x
k

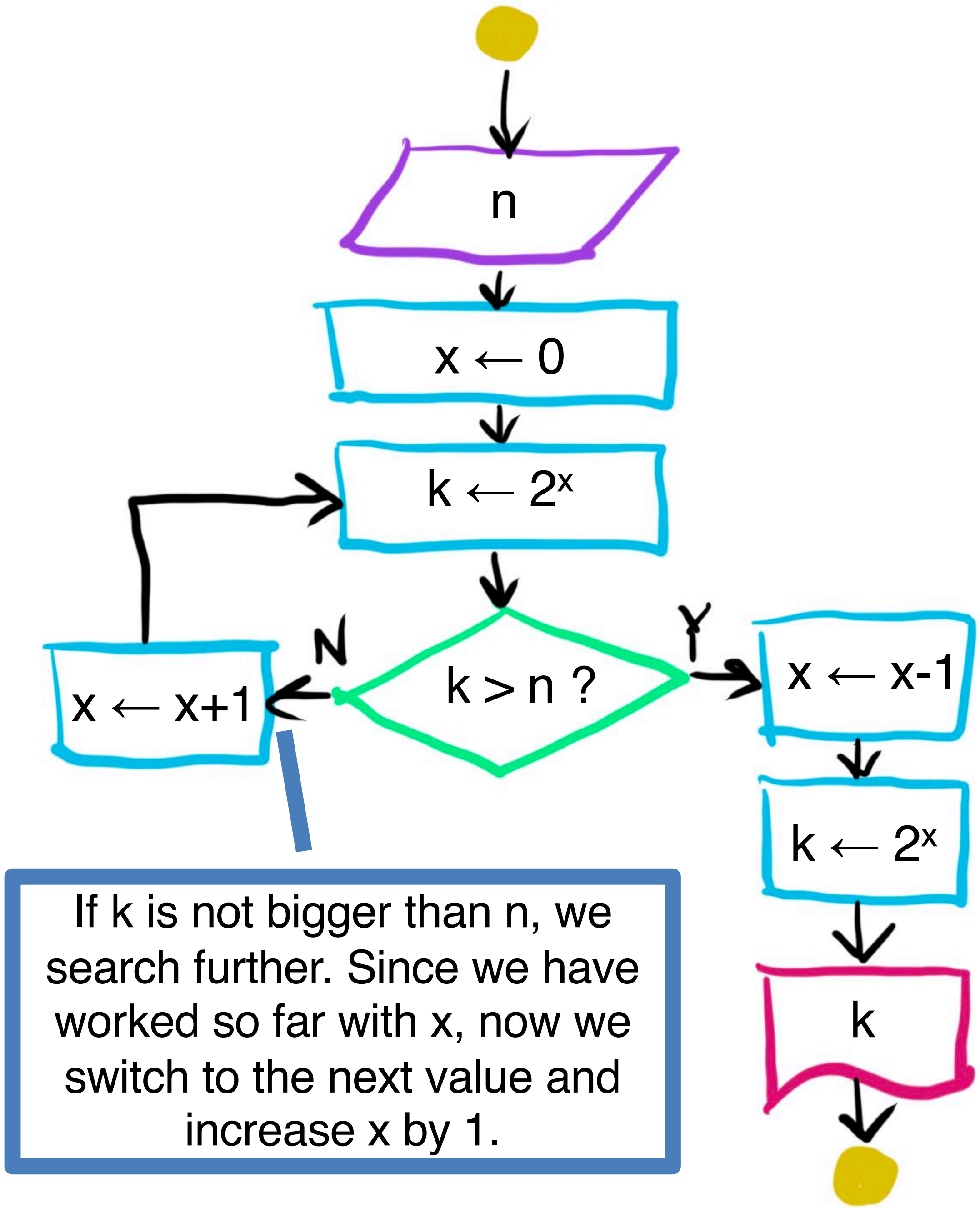

This input is parametric: the algorithm is meant to work for any value n that comes, not just a specific one (like 5).
n
x ← 0
k ← 2^x
k > n ?
N
Y
x ← x+1
x ← x-1
k ← 2^x
k

x is the value of the exponent in the power; since the computer has the search the correct value, the starting value is 0, the smallest one, so that the search does not miss out any value. This is an initialization of x to 0.

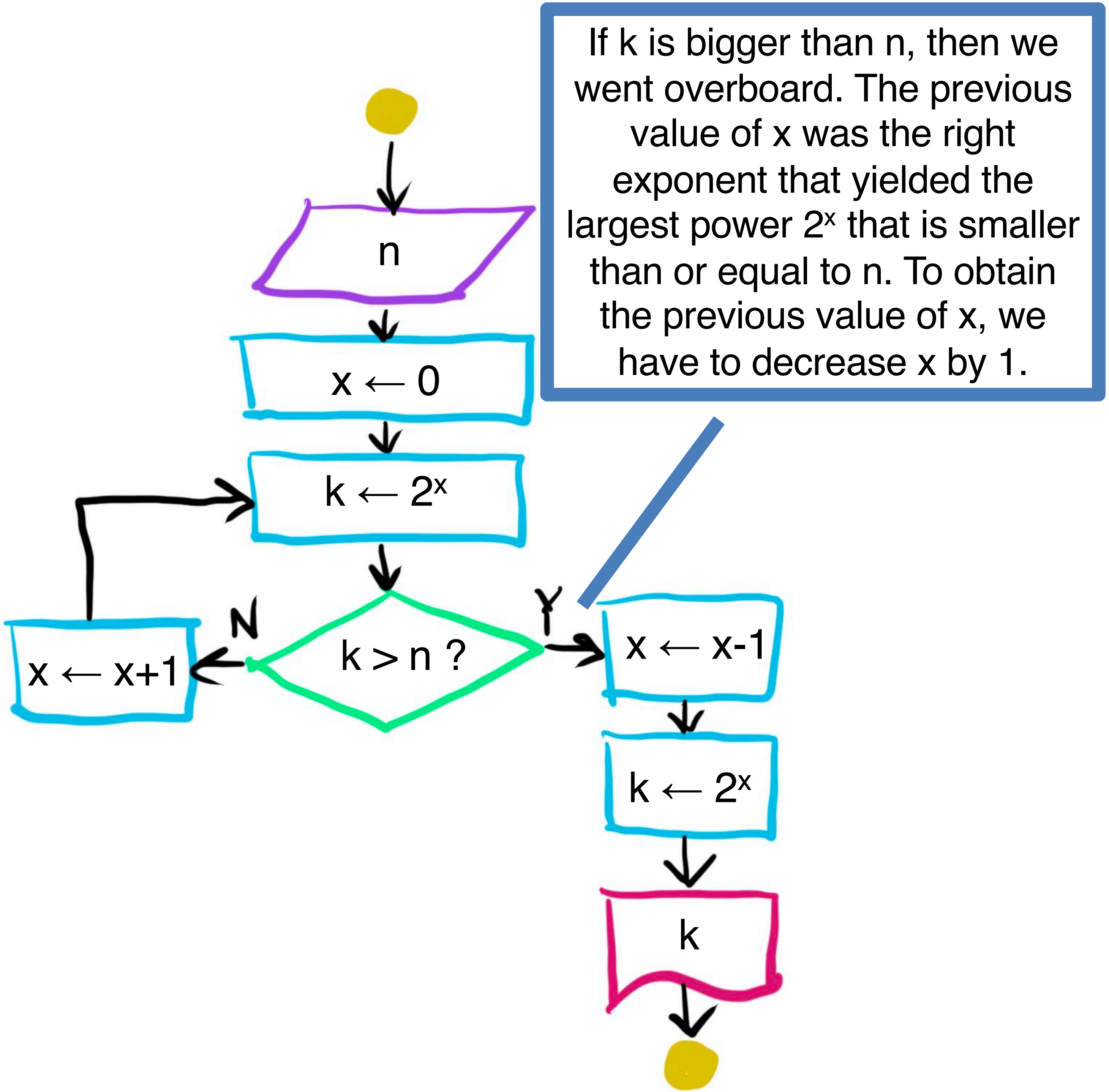




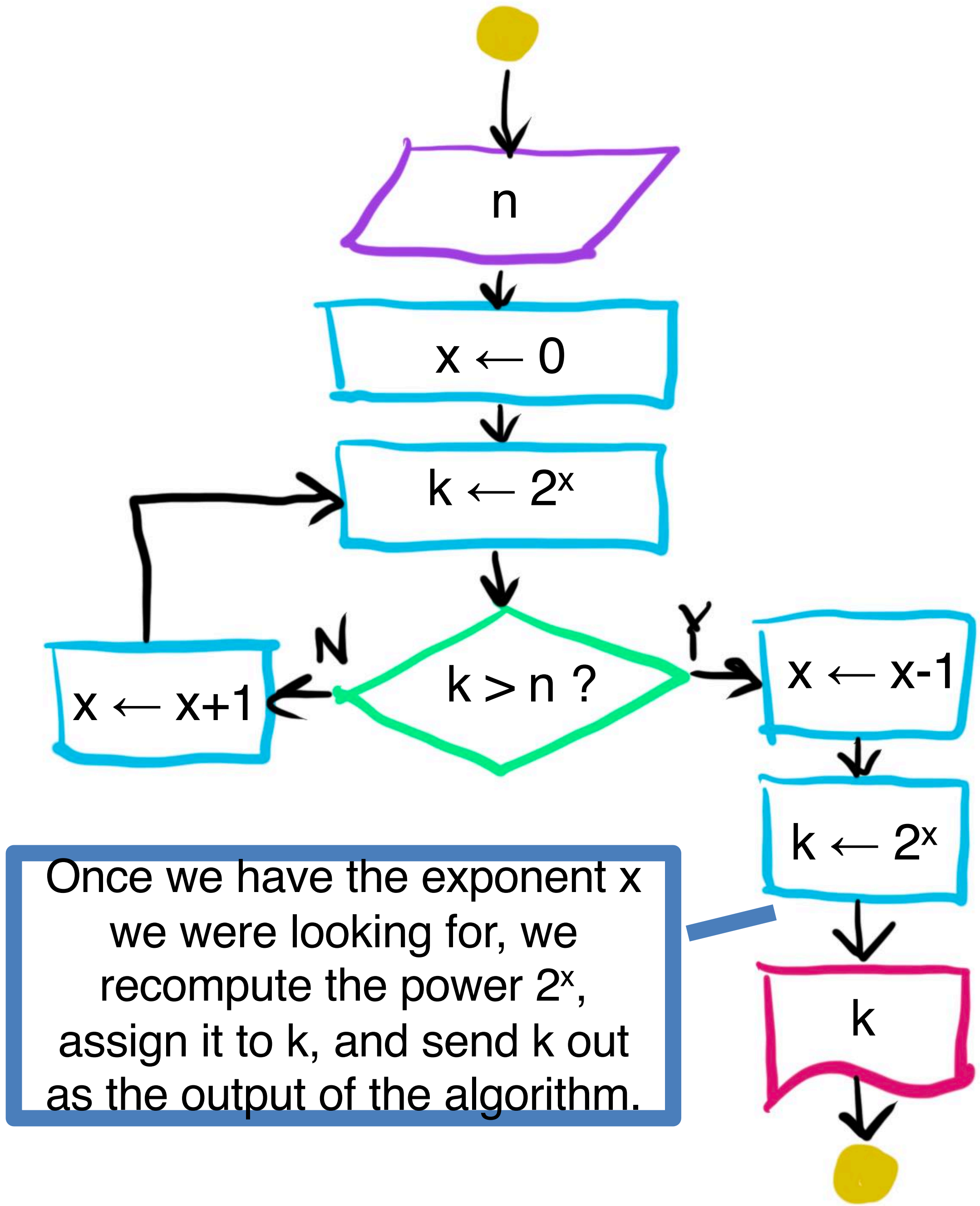
n
x ← 0
k ← 2^x
We compute the power and assign the value of the result to a variable that we call "k" (it could have been any letter except for "n" or "x", which are already used for other purposes).
N
k > n ?
Y
x ← x+1
x ← x-1
k ← 2^x
k



n
x ← 0
k ← 2^x
k > n ?
N
Y
x ← x+1
x ← x-1
k ← 2^x
k
If k is not bigger than n, we search further. Since we have worked so far with x, now we switch to the next value and increase x by 1.



If k is bigger than n, then we went overboard. The previous value of x was the right exponent that yielded the largest power 2^x that is smaller than or equal to n. To obtain the previous value of x, we have to decrease x by 1.
n
x ← 0
k ← 2^x
k > n ?
N
Y
x ← x+1
x ← x-1
k ← 2^x
k


n
x ← 0
k ← 2^x
k > n ?
N
Y
x ← x+1
x ← x-1
k ← 2^x
k
Once we have the exponent x we were looking for, we recompute the power 2^x, assign it to k, and send k out as the output of the algorithm.

Second method:
we divide the number by 2 and obtain quotient and remainder.
As long as the quotient is not 0, we take it as the new dividend and continue dividing. When we get zero quotient, we have to write the list of remainders in reverse order to get the binary encoding of the initial number.
Let's take 144 as an example:

144 : 2 = 72 with remainder 0
72 : 2 = 36 with remainder 0
36 : 2 = 18 with remainder 0
18 : 2 = 9 with remainder 0
9 : 2 = 4 with remainder 1
4 : 2 = 2 with remainder 0
2 : 2 = 1 with remainder 0
1 : 2 = 0 with remainder 1

144 in base 2 is 10010000 (these bits are the remainders in the reverse order of writing)

Exercise for you: draw the flowchart for the algorithm that corresponds to the second method.

Second method:
we divide the number by 2 and obtain quotient and remainder.
As long as the quotient is not 0, we take it as the new dividend and continue dividing. When we get zero quotient, we have to write the list of remainders in reverse order to get the binary encoding of the initial number.
Let's take 144 as an example:

144 : 2 = 72 with remainder 0
72 : 2 = 36 with remainder 0
36 : 2 = 18 with remainder 0
18 : 2 = 9 with remainder 0
9 : 2 = 4 with remainder 1
4 : 2 = 2 with remainder 0
2 : 2 = 1 with remainder 0
1 : 2 = 0 with remainder 1

144 in base 2 is 10010000 (these bits are the remainders in the reverse order of writing)

Encoding of numbers

Numbers with base 10:

$$215 = 2\times10^2 + 1\times10^1 + 5\times10^0$$

Numbers with base 2:

$$101 = 1\times10^2 + 0\times10^1 + 1\times10^0$$

Encoding of numbers

Numbers with base 10:

$$215 = 2\times10^2 + 1\times10^1 + 5\times10^0$$

Numbers with base 2:

$$101 = 1\times2^2 + 0\times2^1 + 1\times2^0$$

Numbers with base 8:

$$723 = 7\times8^2+2\times8^1+3\times8^0 = 467$$

There are also systems where the base is greater than 10, like the “hexadecimal” system, where the base is 16, that is, we have 16 digits:

0 1 2 3 4 5 6 7 8 9 A B C D E F

it corresponds to 10

it corresponds to 15

Numbers with base 16:

$$3AF = 3\times16^2 + A\times16^1 + F\times16^0$$

$$= 3\times256 + 10\times16 + 15\times1$$

$$= 768 + 160 + 15 = 943$$

$$3AF_{16} = 943_{10}$$

Often, in the IT world, they use an alternative notation to n_{16}, that is 0xn (e.g. 0x3AF)