Background: Algorithms

Algorithm is a detailed set of instructions.
Natural languages are rather vague.
To describe an algorithm unambiguously a formal language with precisely defined interpretation must be used.

To solve a problem, express the solution as a very detailed algorithm that can then be converted to a program written in some programming language.

A compiler (another program) will convert the **source code** to **object code** understood by the computer.

Usually some libraries must be linked to the object code to create an **executable program**.

Example 1: Find the largest element in a set of numbers.

This instruction is sufficient for humans, but does not specify what to do:
- what are the numbers?
- how the largest number is found?
- how the result is used?

A mathematical version:

$$x = \max\{a_1, a_2, \ldots, a_n\}$$

Even this does not tell how the largest value is found.

Algorithmic version:

$$
\begin{aligned}
&x = a_1,\\
&\text{if } a_2 > x,\ x = a_2,\\
&\text{if } a_3 > x,\ x = a_3,\\
&\vdots\\
&\text{if } a_n > x,\ x = a_n.
\end{aligned}
$$

This can be programmed if n is nown.

But n can have different values.

A better algorithm:

$$
\begin{aligned}
&n = 100 \\
&x = a_1, \\
&i = 2, 3, \dots n : \\
&\quad \text{if} a_i > x, \ x = a_i
\end{aligned}
$$

The example contains three basic control structures:

1) Sequential execution: First, the number of elements is set as 100,
next, the value of the first number is assigned to x,
finally, the largest element is searched.

2) Choice: If the current element is bigger than the largest element found this far, replace the largest element by the new value.

3) Iteration: Repeat the comparison for all numbers of the set.

This could be written as a program

```
n = 100
x = a(1)
do i=2,n
  if (a(i) > x) x = a(i)
end do
```

This is not yet a complete program.
– Were are the values a_i coming from?
– How the result is used?

Example 2: Sieve of Eratosthenes

A simple method for finding primes.

1. Write down numbers $2, 3, 4, \ldots, n$.

2. Remove the multiples of 2 (4, 6, 8 jne.).

3. The next element in the list is a prime

4. remove numbers that are multiples of the number found in the rpevious step.

5. If the next remaining number is $< \sqrt{n}$, return to step 3.

6. All remaining numbers are primes.

This is already pretty detailed, but step 2 contains iteration that needs some tuning befo-
re it can be programmed.