Algorithms, pseudocode, programming code; overview

1. Algorithm
2. Three presentation levels
3. Pseudocode

1. Algorithm

At start: some problem (questions) we want to solve, given some material/information

At end: solution (answers) to the problem

Definition of algorithm (Levitin A. *The design and analysis of algorithms*)

An algorithm is a sequence of unambiguous instructions for solving a problem i.e. for obtaining a required output for any legitimate input in a finite amount of time.

Example

Start: you are inside with socks on; a (suitable) pair of your shoes exist inside
End: both shoes on correct feet and you are ready to go outside

An algorithm for putting on shoes:

1. Find the left shoe.
2. Find the right shoe.
3. Put the left shoe on your left foot.
4. Put the right shoe on your right foot.
5. If your left shoe has laces and the laces need to be tied, then tie the laces.
6. If your right shoe has laces and the laces need to be tied, then tie the laces.
7. If your left shoe has straps and the straps need to be fasten, then fasten them.
8. If your right shoe has straps and the straps need to be fasten, then fasten them.
9. Your shoes are on.

□

In computer science:

Algorithm starting point: problem with well specified starting (input) data X

Algorithm ending point: well specified output data Y that solves problem

Algorithm: how do we compute Y from X?

Unambiguous instructions:

- each instruction is given in sufficient detail so that the device performing the algorithm cannot misunderstand it

- each instruction has only one interpretation

- the order of the instructions has only one interpretation, which cannot be misunderstood

- the conditions for terminating the algorithm cannot be misunderstood (algorithm must terminate!)

Legitimate input

- what input is acceptable/unacceptable must be well specifed

- what form/format the input can be given in must be well specifed

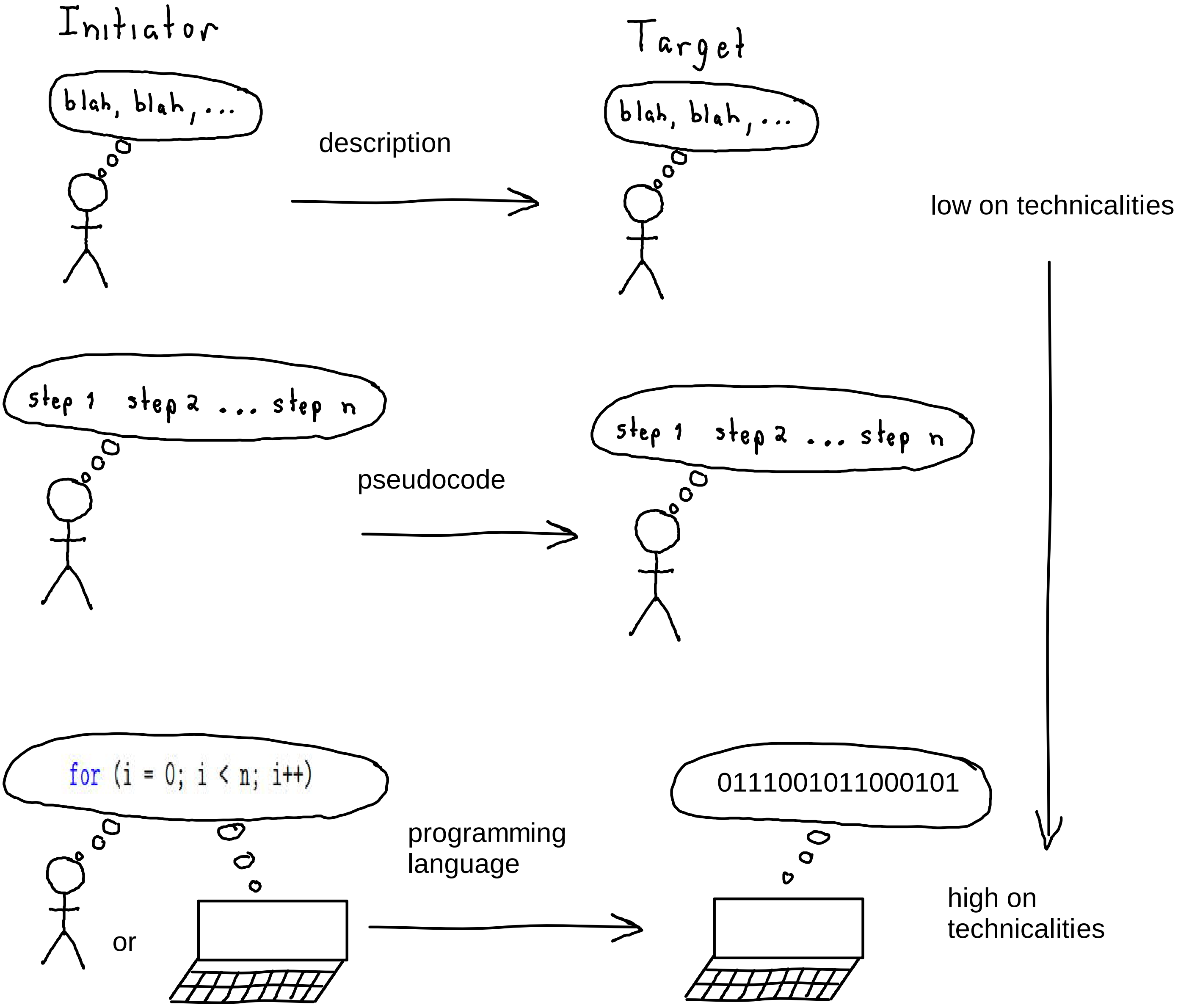
Finite amount of time

- algorithm must terminate!
- conditions under which algorithm terminates must be well specified

2. Three presentation levels

Three levels of presentation for some algorithm:

- description level: person to person
- pseudocode level: technical person (programmer) to technical person (programmer)
- program code level: programmer/computer to computer



Description level

- describe what each step/stage of algorithm does
- can be mixture of ordinary language, math, very simple code

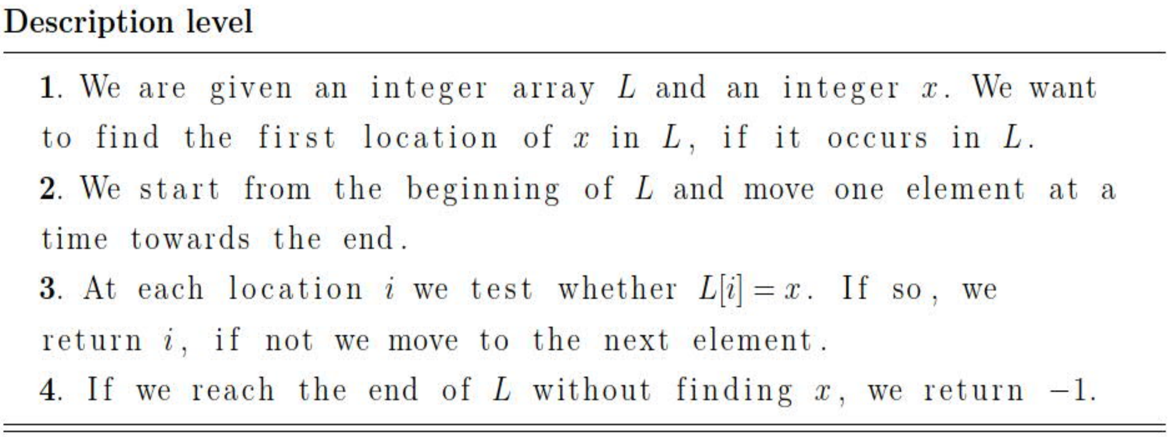
Pseudocode level

- not a programming language ready for compiling and/or execution
- parameters and variables used and defined
- use typical control structures and blocks, e.g. if-then-else-end, for-end, while-end, for each-end
- no error handling, checking on inputs, modularity, etc.
- avoid program language specifics e.g. ++ for incrementaion of integer counter

Progam code level

- some recognized language C++, Java, Python, Matlab, etc.
- intended for compilation/execution
- syntax requirements
- other specification must be met: error handling, checking validity of input, etc.

Example



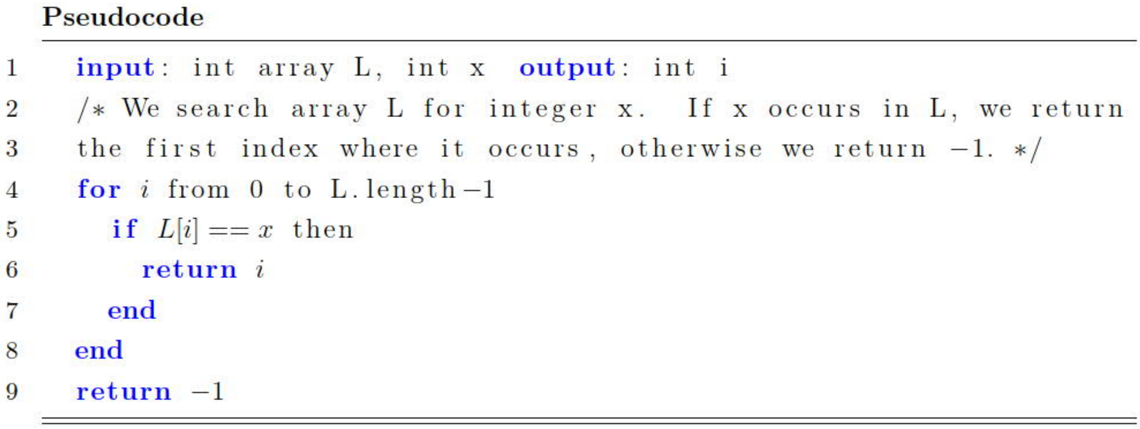
Description level

1. We are given an integer array L and an integer x. We want to find the first location of x in L, if it occurs in L.

2. We start from the beginning of L and move one element at a time towards the end.

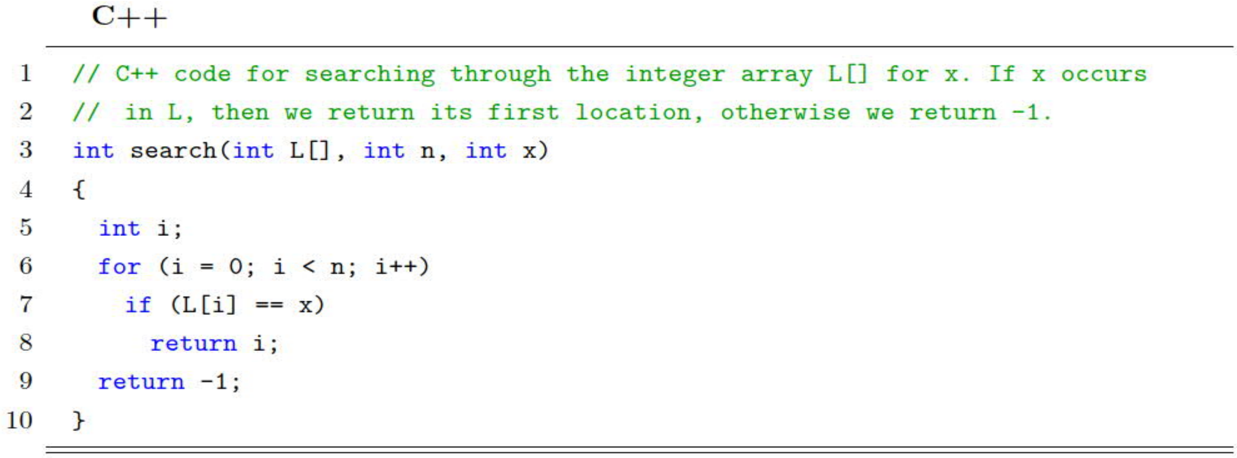
3. At each location i we test whether $L[i] = x$. If so, we return i, if not we move to the next element.

4. If we reach the end of L without finding x, we return -1.



Pseudocode

```
input: int array L, int x  output: int i
/* We search array L for integer x.  If x occurs in L, we return
the first index where it occurs, otherwise we return −1. */
for $i$ from 0 to L.length−1
  if $L[i] == x$ then
    return $i$
  end
end
return −1
```



C++

```cpp
// C++ code for searching through the integer array L[] for x. If x occurs
//  in L, then we return its first location, otherwise we return -1.
int search(int L[], int n, int x)
{
  int i;
  for (i = 0; i < n; i++)
    if (L[i] == x)
      return i;
  return -1;
}
```

□

3. Pseudocode

Suppose Jussi knows Finnish, English, Java and python.

Suppose Mio knows Japanese, Korean, C++ and C#.

How can they communicate efficiently an algorithm?

What we can ignore:

- error handling
- variable types (often)
- file I/O, formatting
- syntax (to a certain extent)
- hardware
- programming language

What we can focus on:

- algorithm computations/operations
- essential variables/parameters

Pseudocode specifications

- indentation is used to indicate blocks
- control-blocks similar to C++:

  if-then-else-end, for-end, while-end,

- control block used to iterate through elements in a set/list

  for each-end

- comments within “/* */”

- assignment made with single equals “=”, testing made with double equals “==”

- variables are local to a given procedure (function)

- element in i’th location in array L is L[i]

- elements in locations i, i + 1, . . . j in array L are L[i..j]

- an object contains compound data; an object has attributes, e.g.

  person : object name
  person.firstname : one object attribute
  person.lastname : another object attribute

- return transfers control back to calling procedure immediately

- NIL is a reference (pointer) to nothing

- use 8 to present a free-form instruction, e.g.

  8 check that all numbers in array L are even

These specifications are **not** universal!

tekijä: Frank Cameron

made by Frank Cameron