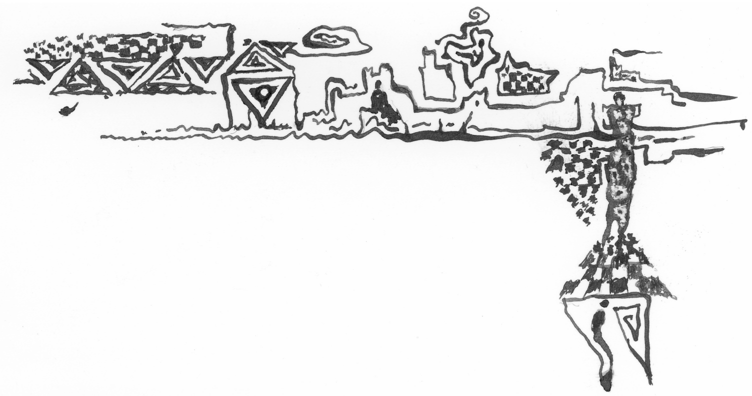
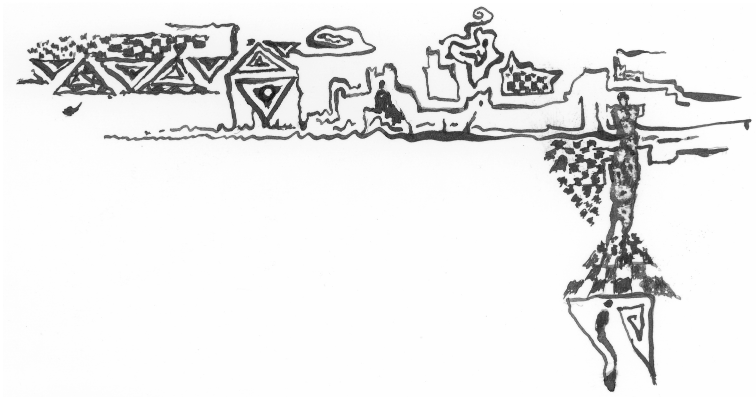
Algorithmic Adventures

From Knowledge to Magic



Book by Juraj Hromkovič, ETH Zurich
Slides by Tom Verhoeff, TU Eindhoven

Quotation



Perfection is based upon small things,
but perfection itself is no small thing at all.

Michelangelo Buonarroti

Algorithmic Cooking

An algorithm provides simple and unambiguous advice on how to proceed step by step in order to reach a specified goal.

To what extent may one view a cooking recipe as an algorithm?

Ingredients for apricot flan:

```
    3   egg whites
    1   pinch of salt
    6   tablespoons of hot water
100 g   cane sugar
    3   egg yolks
    1   teaspoon of lemon peel
150 g   flour
  1/2   teaspoon of baking powder
400 g   peeled apricots
```

Recipe for apricot flan:

```
1. Put greaseproof paper into a
   springform pan!
2. Heat the oven up to 180°C!
3. Heat up 6 tablespoons of
   water!
4. Mix three egg whites with
   the hot water and a pinch of
   salt, beat them until you get
   whipped egg white!
... ...
```

Cooking Recipe

4.6 Mix the content of G for 2 minutes.

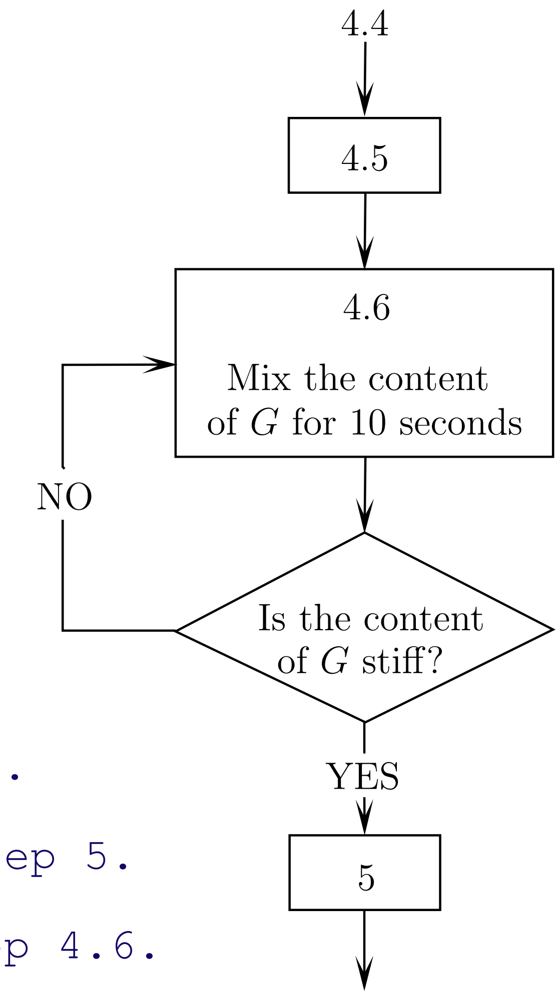
Refine this as:

4.6 Mix the content of G for 10 seconds.

4.7 Test whether the content of G is stiff or not.

If the answer is "YES", then continue with step 5.

If the answer is "NO", then continue with step 4.6.



Flowchart

Computer Algorithms

- Fix a list of fundamental instructions (operations) that a computer can execute without any doubt.

- Possible (abstract) inputs: (infinitely many) **problem instances**

- Required output solving the problem

  An algorithm for solving a problem (a task) has to ensure that it works correctly for each possible problem instance.

  To work correctly means that, for any input, it finishes its work in a finite time and produces the correct result.

Computer Programs and Programming

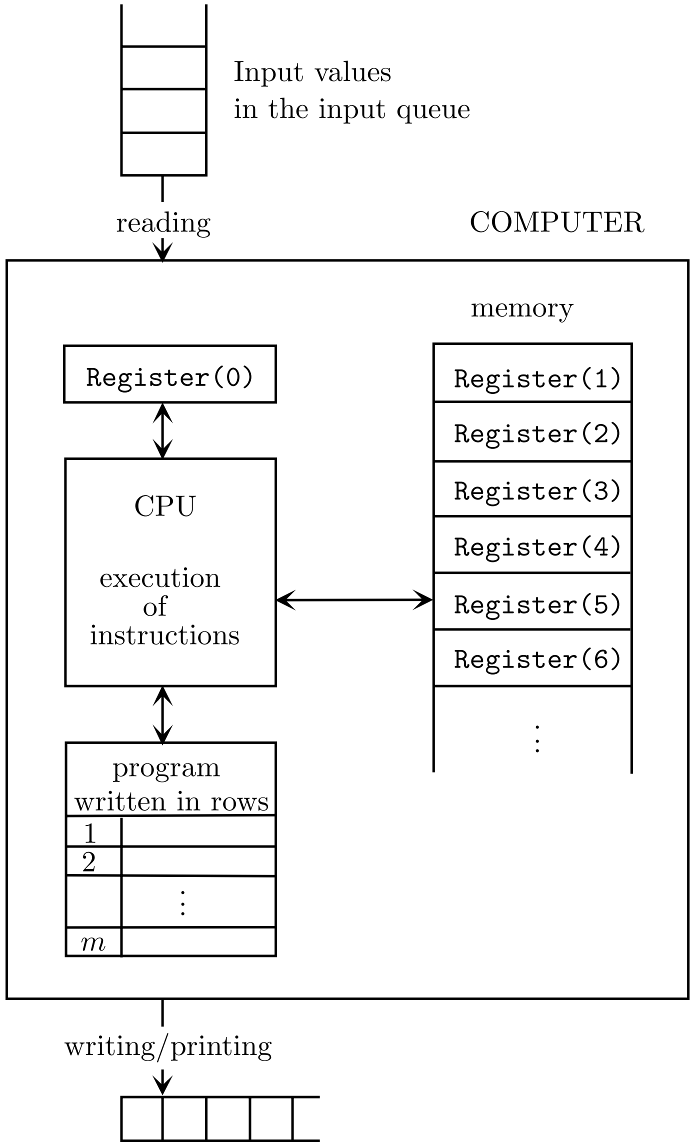
A **program** is a *sequence of computer instructions* that is represented in a form understandable by a computer.

1. A program does not need to be a representation of an algorithm. A program may be a meaningless sequence of computer instructions.

2. An algorithm need not be written in the form of a program. An algorithm can also be described in a natural language or in the language of mathematics. A program must be expressed in a special formalism of the given **programming language**.

Programming is the *activity of rewriting algorithms* into programs.

Computer Model



Computer Model (Terminology)

- A **memory** that consists of a large number of memory cells, called **registers**, numbered by positive integers, called **addresses** of the registers. Each register can save an arbitrarily large number.

- A special memory in which the whole program is saved. Each line of the program consists of exactly one instruction of the program. The lines are numbered starting at 1.

- There is a special register `Register(0)` that contains the number of the just executed instruction (line) of the program.

- A CPU (central processing unit) that is connected to all other parts of the computer, doing all the work.

Programming Language ‘TRANSPARENT’, Part 1

(1) `Read into Register(`n`).`

(2) `Register(`n`)` $\leftarrow k$

(3) `Register(`n`)` $\leftarrow$ `Register(`j`) + Register(`i`)`

(4) `Register(`n`)` $\leftarrow$ `Register(`j`) - Register(`i`)`

(5) `Register(`n`)` $\leftarrow$ `Register(`j`) * Register(`i`)`

(6) `Register(`n`)` $\leftarrow$ `Register(`j`) / Register(`i`)`

(7) `Register(`n`)` $\leftarrow \sqrt{\texttt{Register}(m)}$

Programming Language ‘TRANSPARENT’, Part 2

(8) `If Register(`n`) = 0, then go to row` j

(9) `If Register(`n`)` $\leq$ `Register(`m`), then go to row` j

(10) `Go to row` j

(11) `Output` $\leftarrow$ `Register(`j`)`

(12) `Output` $\leftarrow$ “`Text`”

(13) `End.`

(14) `Register(Register(`i`))` $\leftarrow$ `Register(`j`)`

Never-Ending Execution

One of our most important demands on the definition of an algorithm for a computing task is that the algorithm finishes its work for any input and provides a result.

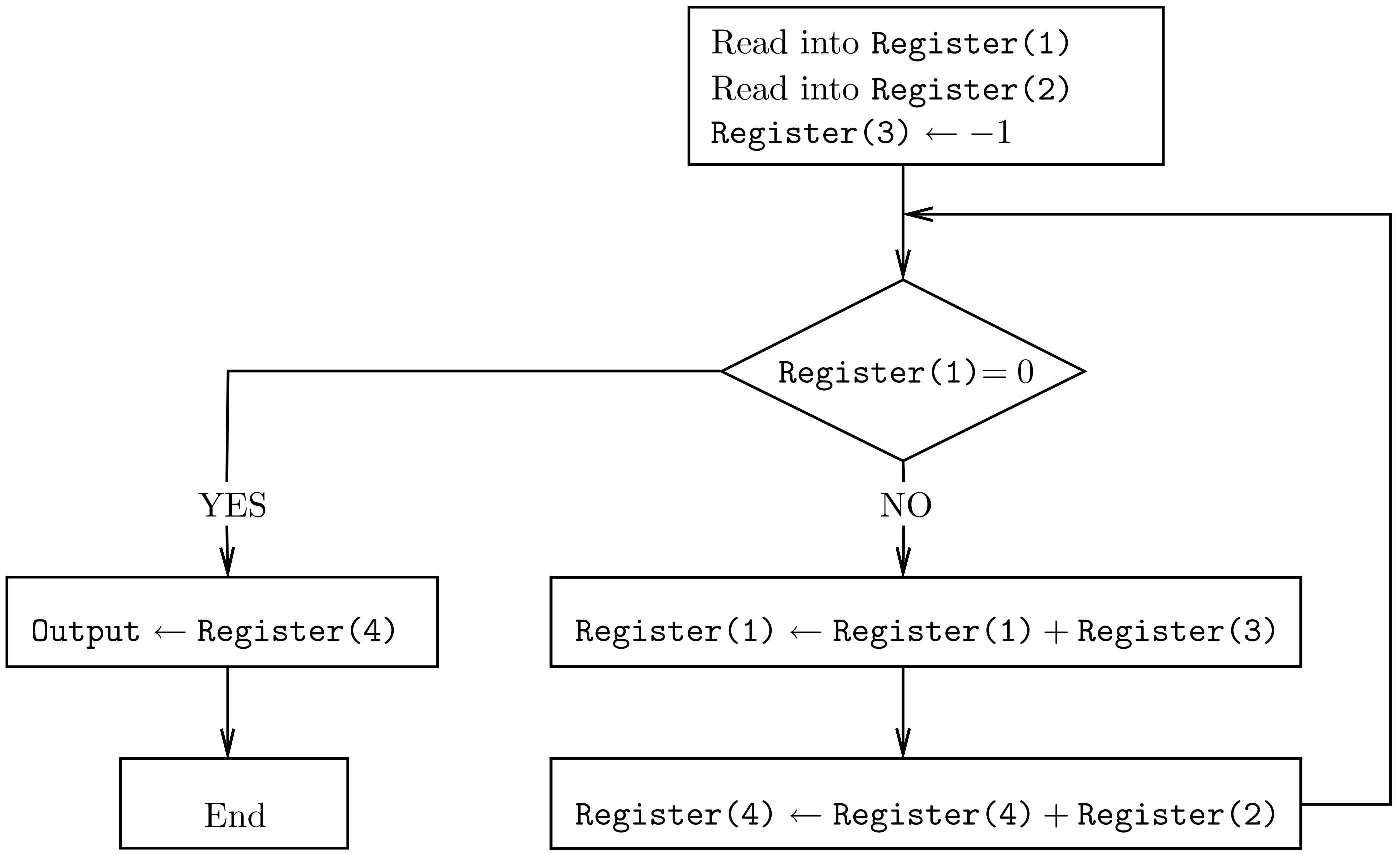
In the formal language of computer science, we speak about **halting**.

If an algorithm A finishes its work on an input (a problem instance) in a finite time, then we say that **algorithm A halts on x**.

In this terminology, we force a halt of the algorithm on every possible input and in such a case we say that A **always halts**.

A program can engage in never-ending execution.

Example



What does this program compute? What if `Register(1)` starts < 0?

Summary

- One has to be able to *apply* an algorithm even if one is not an expert in solving the considered problem. One does not need to understand *why* the algorithm provides the solution of the problem. It suffices to be able to execute the simple activities the algorithm consists of.
- Defining the notion of an algorithm, one has to list all such *simple activities* and everybody has to agree that all these activities are executable by a machine.
- An algorithm is designed not only to solve a problem instance, but it must be applicable to solving *all* possible instances of a given problem.
- We require a guarantee that an algorithm for a problem successfully finds a solution for *each* problem instance.