Programming is not Coding

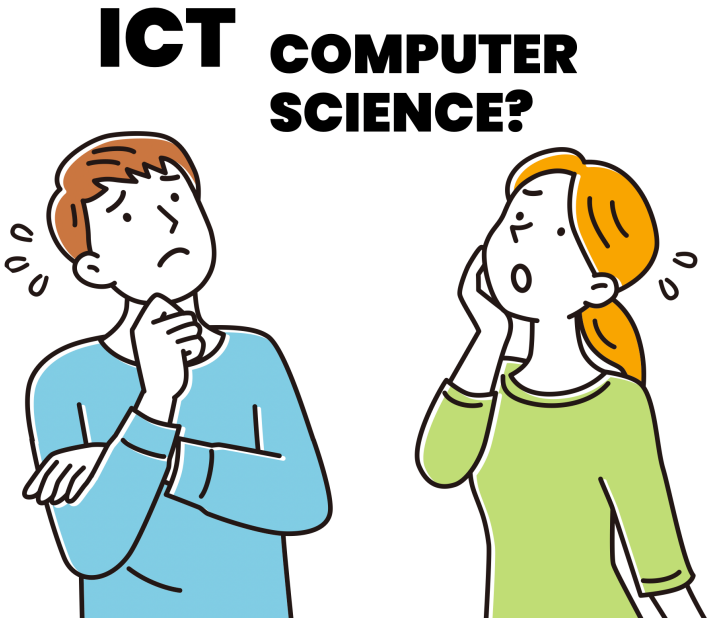
How to build safe and secure software

Stefan Parvu

November 7, 2023

Lesson 1
Computation. Algorithms. Programs

In this lesson we will talk about the following concepts: computation, algorithms and programs.


ICT
COMPUTER SCIENCE?

Computer Science (CS)

But what's the difference?

Computer Science	Information Technology
Computer Science is the study of the principles of computing and how computer systems solve problems	IT is the study of current tools and computing techniques that can be used for technological needs of a particular organization
Computer Scientists design and build tools and software applications	IT professionals apply and use these tools or software applications
Programming computers using mathematical algorithms, including abstract concepts like computational complexity theory	IT involves more practical aspects how to use and maintain software applications to solve and develop new business processes

Computation

What is a Computation?

A **computation** is any type of arithmetic or non-arithmetic calculation that is well-defined.[1][2] Common examples of computations are mathematical equations and computer algorithms.

Mechanical or electronic devices (or, historically, people) that perform computations are known as *computers*. The study of computation is the field of computability, itself a sub-field of computer science.

A **computation** is what a computing device does, no matter of the computing device: a computer system, a tablet, a mobile phone.

Example of computations: well-defined mathematical statements, which are solvable. But there are mathematically concepts which are not well-defined, and can be difficult to solve, like: the halting problem, or busy beaver game.

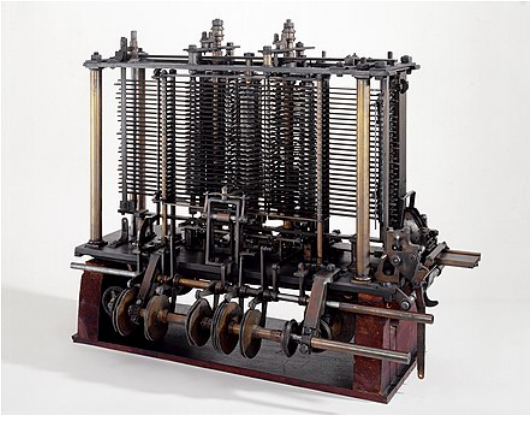
Computing devices are supposed to compute something. Like calculate and predict the weather, render to produce a movie, browse the Internet.

Some well-defined computations:

- calculations carried by an electronic computer or calculator
- calculations carried out on an analytical engine
- majority of mathematical statements and calculations

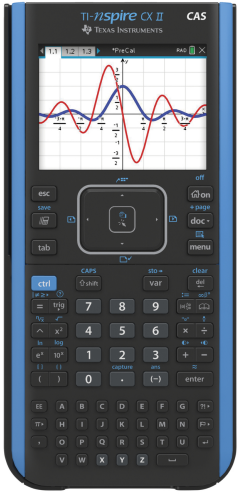
What is an Analytical Engine?

- The very first mechanical general-purpose computer
- Designed by the English mathematician and computer pioneer Charles Babbage
- Designed in 1837



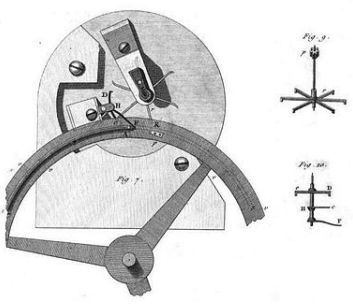
The basic calculator

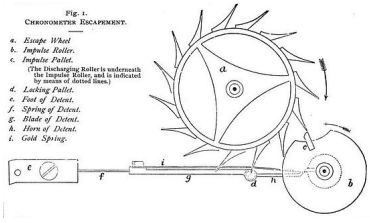
- Execute a number of precise operations
- Designed to contain a set of such operations and instructions
- Includes even more complex operations, graphing charting

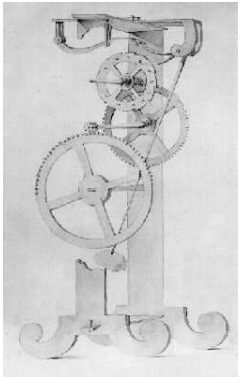


World's first computing device?

The escapement clock, a man-made device, to keep track of time.







Computation can be seen as a purely physical process occurring inside a closed **physical system** called a **computer**.

Example of physical systems: analytical engine machines, human mathematicians following strict rules, digital computers, mechanical computers, analog computers and others.

Algorithms

What is an algorithm?

In mathematics and computer science, an **algorithm** (/ˈælgərɪðəm/) is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.[1] Algorithms are used as specifications for performing calculations and data processing.

- But can we call this a computer program?
- Or is it just a recipe, something higher than a sequence of code
- A higher level of abstraction of how to implement something we plan to develop. Example: how to find a name in a phone book

Algorithms are everywhere

From your kitchen, in your microwave oven, your washing machine, to your phone or computer. When you browse Internet web sites your web browser is using different algorithms to decide how to display data to you.

Our society relies on algorithms to suggest sentences for convicted criminals. You even use algorithms to keep you alive: the control systems from your car, or in different medical devices.

But how can we describe them?

As an abstraction of something you plan to build or use, including all basic operations to achieve that. For example, think you plan to search in a phone book, a person phone number by the name:

- you can start page by page searching for that name
- or you can jump directly to certain letters and start following from there
- or you can apply a different strategy, by 'cutting' the book into half, checking the letter in which half belongs, and applying all over again the same principle until the name is found

How can we write one algorithm?

You can write it in plain English or in a more precise way using mathematics. Some others are using a form of pseudocode.

In computer science, **pseudocode** is a description of the steps in an algorithm using a mix of conventions of programming languages (like assignment operator, conditional operator, loop) with informal, usually self-explanatory, notation of actions and conditions.[1][2] Although pseudocode shares features with regular programming languages, it is intended for human reading rather than machine control. Pseudocode typically omits details that are essential for machine understanding of the algorithm. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation.

Example phonebook

1: **procedure** PHONEBOOK(N) ▷ Returns person name: N
2: $N \leftarrow 1$
3: Open page number N
4: Look at the page N
5: **if** Person is on page N **then**
6: **Call person** N ▷ The person name is N
7: **else**
8: **Find next page** $N \leftarrow$N $+ 1$ Go back to 3
9: **end if**
end procedure

But why not using a programming language?

We must think what we are planning to do, and how we plan to do it. And for that, we must not rely on a programming language: we will be restricted by the limits of the specific programming language not being able to design and think freely about it.

A better approach is to think to write it as pseudocode or simple mathematics. This way we can have all flexibility and the power of precise mathematics.

Euclid's Algorithm or GCD

In mathematics, the **Euclidean algorithm**,[note 1] or **Euclid's algorithm**, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder. It is named after the ancient Greek mathematician Euclid, who first described it in his *Elements* (c. 300 BC). It is an example of an *algorithm*, a step-by-step procedure for performing a calculation according to well-defined rules, and is one of the oldest algorithms in common use. It can be used to reduce fractions to their simplest form, and is a part of many other number-theoretic and cryptographic calculations.

Euclid's Algorithm

Greatest Common Divisor (GCD) of two numbers A and B is the largest number that divides both A and B. (Number here defined as an integer)

An integer is the number zero (0), a positive natural number (1, 2, 3, etc) or a negative integer with a minus sign (-1, -2, -3, etc) In mathematics we call this $\mathbb{Z}$ set of numbers.

Euclid's Algorithm

The very first version:

If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and STOP
If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and STOP
Write A in quotient remainder form (A = B * Q + R)
Compute then GCD(B,R) since GCD(A,B) = GCD(B,R)

Euclid's Algorithm, pseudocode

```
1: procedure GCD(A, B)                    ▷ The g.c.d. of A and B
2:     R ← A mod B
3:     while R ≠ 0 do                     ▷ We have the answer if R is 0
4:         A ← B
5:         B ← r
6:         R ← A mod B
7:     end while
8:     return B                           ▷ The gcd is B
9: end procedure
```

Euclid's Algorithm, improved pseudocode

```
1: procedure EUCLID(A, B)                    ▷ The g.c.d. of A and B
2:     if B == 0 then
3:         return A                          ▷ The gcd is A
4:     else
5:         return EUCLID(B, A mod B)
6: end if
end procedure
```

Euclid's Algorithm, using mathematics

MODULE *Euclid*

EXTENDS Integers

VARIABLES x, y

CONSTANTS a, b

$$Init \triangleq (x = \text{a}) \land (y = b)$$

$$\begin{aligned} \text{Next} \triangleq\ & (\ x > y \\ & \land x' = x - y \\ & \land y' = y\ ) \\ \lor\ & (\ y > x \\ & \land y' = y - x \\ & \land x' = x\ ) \end{aligned}$$

Classes of algorithms

- Divide and Conquer
- Sorting
- Searching
- Dynamic programming
- Greedy algorithms
- Graph algorithms
- Shortest path
- Maximum flow
- Parallel algorithms
- Matrix operations
- Online algorithms
- Machine learning
- Linear programming
- String matching

Computer Programs

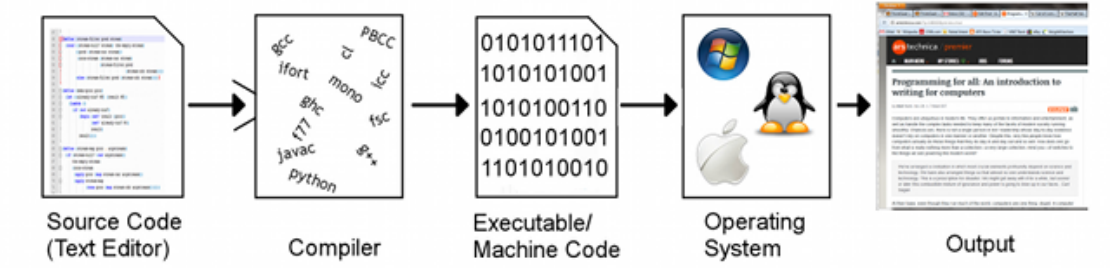
What is a computer program?

A **computer program** is a sequence or set of instructions in a programming language for a computer to execute. It is one component of software, which also includes documentation and other intangible components.[1]

- A computer program in its human-readable form: source code
- The source code needs to be translated into machine code to be able to execute
- The resulting file is called an executable file
- Then the operating system loads the executable into memory and starts a process to execute it

Computers, digital systems are executing programs

From source code to executable



Program structure

A **computer program** is a sequence or set of instructions in a programming language for a computer to execute. It is one component of software, which also includes documentation and other intangible components.[1]

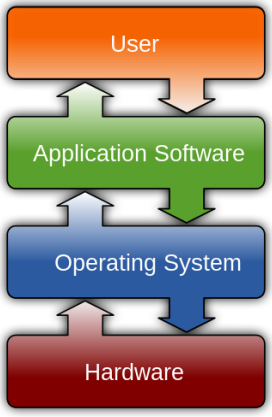
- Some programs run forever, some dont
- A program execution is defined by at least one computation
- A computation is a sequence of states
- And a state is an assignment of values to variables

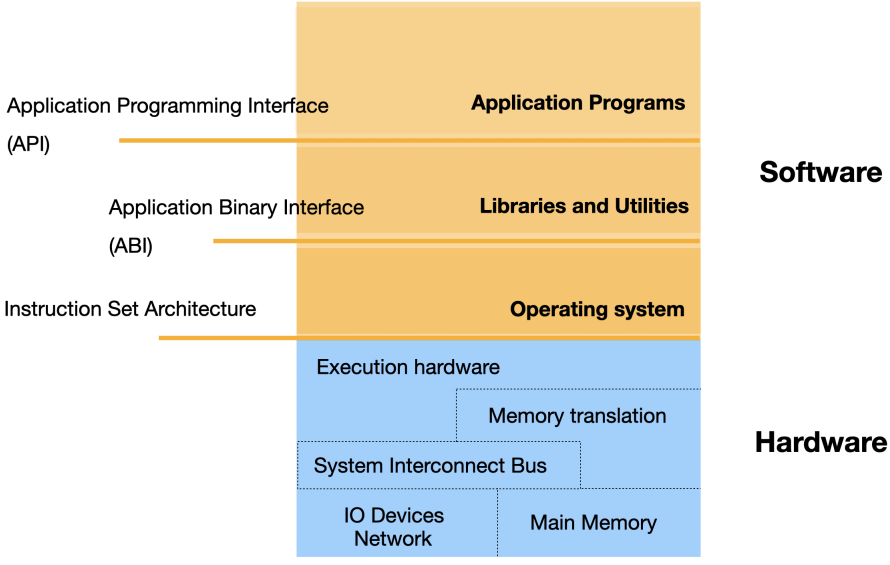
Program types

- A program is modelled by a set of computations, representing all possible executions
- Remember an algorithm is just an abstract program
- Different programs: software applications and system software
- System software: operating systems

Application and System Programs

- **Software applications**: enterprise resource planning, customer relationship management, supply chain management software, web, middleware, databases
- **Operating systems**: macOS, RedHat, FreeBSD, Windows




Application Programming Interface
(API)
Application Binary Interface
(ABI)
Instruction Set Architecture
Application Programs
Libraries and Utilities
Operating system
Execution hardware
Memory translation
System Interconnect Bus
IO Devices
Network
Main Memory
Software
Hardware