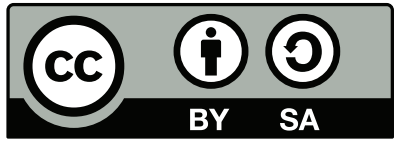
Laboratory of Computer Science Education

Didattica della programmazione

Alberto Montresor

Università di Trento

2022/04/02



Cosa significa saper programmare?

(du Boulay, 1989)

- General orientation, what programs are for and what can be done with them;
- the Notional machine, a general model of the computer as it relates to executing programs;
- Notation, the syntax and semantics of a particular programming language;
- Structures, the use of schemata / plans as ways of organising knowledge;
- Pragmatics, the skills of planning, developing, testing, debugging, and so on

Cosa significa saper programmare?

(Bayman and Mayer 1988) (McGill and Volet 1997)

- Syntactic knowledge is the knowledge of the language features, basic facts, and rules.
  - Example: double quotation marks, semi-colon, syntax of "for"
- Conceptual knowledge refers to the knowledge of how programming constructs and principles work and what happens inside the computer.
  - What happens behind an assignment statement
  - What happens inside a loop
- Strategic knowledge refers to how to apply syntactic and conceptual knowledge to solve novel problems.
  - Planning, problem solving
  - Developing
  - Testing, debugging

Cosa significa saper programmare? (Monga et al. 2019)

There are two distinct but tightly tied aspects in programming:

- the program itself (the text of the code),
- the actions that take place when the program is run by the interpreter

The first aspect, that is the program source code, is explicit, visible. The second one instead, that is the actions that take place when the program is run, is somewhat implicit, hidden in the execution time world, and not so immediate to grasp for novices.

Moreover, this aspect is sometimes underestimated by both teachers and learners: teachers, as experts, give it for granted; learners tend to construct personal intuitive, not necessarily coherent, ideas of what will happen.

Cosa significa saper programmare? (Monga et al. 2019)

This dichotomy of programming — its static visible code and its implicit dynamics — emerges as a critical issue when learning to program, as shown by studies from different perspectives.

- ... novice programmers tend to perceive programming as no more than the production of code, missing to relating instructions in the program to what happens when the program is executed.
- ... most of programming misconceptions have to do with aspects that are not readily visible in the code but are related to the execution time, both in term of what will happen and of what will not unless explicitly specified in the code.
- ... program dynamics is a candidate threshold concept in programming as it has many of the features that characterise threshold concepts; among others: it is a troublesome barrier to student understanding, it transforms how the student perceives the subject...

Failure rates, fragile learning, bimodal outcomes

- CS1 has typically been regarded as difficult for students, with persistent and widespread reports of high student fail and dropout rates
- Concerns have also been raised about whether all of those who pass CS1 have learned what they should
  `https://runestone.academy/runestone/books/published/StudentCSP/CSPIntroData/rainfall.html`
- The term "bimodal" is often used to describe the resulting grade distributions (with higher than usual rates of both failure and of high grades there are necessarily fewer students in the mid range).

Constructionism and learning to program

Una citazione presa da (Monga et al. 2019)

In some sense, programming is intrinsically constructionist as it always involves the production of an artificat that can be shown and shared. Of course when teaching programming, this aspect can be stressed or attenuated.

- motivation: programming tasks should be engaging to keep pupils' motivation high
- syntax: novices should be introduced first to the logical aspects of programming and only at a later stage to the syntax;
- a constructivist approach: the construction of knowledge is to be fostered, for example through unplugged activities that are more suitable to group work and shared meta-cognition;

Geek gene

Alcune citazioni prese da (Robins 2019)

- Programming is an inherently creative or "artistic" process, that it is not amenable to "scientific" analysis or standard managerial methods
- Frustration at the failure of early attempts to predict aptitude lead to the widely held and subsequently enduring belief that programmers are "born and not made" (Dauw, 1967; Webster, 1996), or (tongue in cheek) that there exists a "geek gene" for programming (Lister, 2010): either you have it or you don't

Discussione

Geek gene (Robins 2019, Section 3.7)

Learning Edge Momentum (LEM) hypothesis

Given some target domain of concepts to be learned, successful learning makes it somewhat easier to acquire related concepts, and unsuccessful learning makes it somewhat harder. In other words, the early acquisition (or otherwise) of concepts in a new domain becomes self-reinforcing, creating momentum towards successful or unsuccessful outcomes.

LEM varies depending on the subject domain

This LEM effect will vary in strength depending on the extent to which the concepts in the target domain are either independent or interdependent. When the domain consists of tightly integrated concepts the momentum effect will be strong.

Geek gene (Robins 2019, Section 3.7)

Programming is hard (Robins 2010)

Further proposed that a typical programming language is a domain of concepts which are unusually tightly integrated (at one end of the spectrum when compared to other domains.

Bimodality is true, but artificial

The tightly integrated nature of language concepts results in a strong LEM effect. It is not the case that programming is simultaneously both hard and easy to learn for two different populations, rather programming effectively becomes both harder and easier to learn for two different emerging groups.

Geek gene (Robins 2019, Section 3.7)

What we have learned?

Studies that have supported predictions of the LEM account of programming outcomes, particularly with respect to the importance of the first 1 to 3 weeks of a CS1 course, include Porter and Zingaro (2014), Porter, Zingaro and Lister (2014), Hola and Andreae (2014) and McCane et al. (2017).

Imparare a programmare - e voi?

Discussione

Vorrei che rifletteste sulla vostra esperienza di apprendimento della programmazione.

- Come è andata nel primo periodo?
- C'è stato un momento "di svolta?"
- Quali sono state le problematiche più importanti?
- Avete osservato i vostri colleghi?

Imparare a programmare - framework

Nell'articolo

- Notional machines (03/04)
- Misconceptions (07/04)
- Abstract programming patterns (10/04)
- Programming language choice
- Learning to program in teams
- Computer science unplugged (28/04)

Alcune cose mancanti

- Problem solving
- Programming paradigms
- ....

Creative learning spyral – Agile development