Di cosa parliamo quando parliamo di ‘programmi’*

Violetta Lonati[1], Claudio Mirolo[2] e Mattia Monga[1]
[1]Università degli Studi di Milano
[2]Università di Udine

Sommario

Il mondo della scuola si sta ormai convincendo che la programmazione debba avere un ruolo sempre più rilevante tra le competenze da acquisire a tutti i livelli e in tutti i percorsi formativi. Del resto è assai opportuno che una parte sempre più ampia della cittadinanza sia in grado di capire cosa significa progettare e realizzare elaborazioni automatizzate. Il rischio, tuttavia, è che la complessità tecnologica spinga a banalizzare gli obiettivi formativi o a soffermarsi su aspetti di dettaglio, perdendo di vista la ricchezza concettuale che la programmazione può dispiegare una volta colte le sue molteplici sfaccettature. Una chiara esposizione degli aspetti chiave dei programmi può aiutare insegnanti e altri operatori culturali a identificare le ragioni della centralità del *software* nella società attuale e a orientare al meglio l’azione educativa, affinché la pratica della programmazione dischiuda tutte le sue potenzialità come strumento di consapevolezza e cittadinanza attiva.

1 La programmazione nella scuola

La programmazione degli “elaboratori di informazioni” sta al cuore dell’informatica. In un certo senso è proprio la ragion d’essere della disciplina, il suo nucleo caratterizzante e motivante l’enorme varietà di approcci e sviluppi che ne vivacizzano lo studio e la pratica. Non stupisce quindi che in una società dominata dalle tecnologie dell’informazione, la programmazione si stia diffondendo sempre di più anche nelle attività scolastiche, a tutti i livelli e in tutti i percorsi formativi [4, 2]. In realtà c’è forse da chiedersi come mai solo ora si inizi ad interrogarsi seriamente su come introdurre compiutamente l’informatica e la programmazione nei percorsi di formazione degli insegnanti. La

*Le riflessioni riportate in questo articolo sono frutto del confronto e della discussione maturata all’interno del gruppo di lavoro WG5, nell’ambito dell’International Conference on Innovation and Technology in Computer Science Education (ITiCSE 2022) di cui fanno parte, oltre ai presenti autori, Tim Bell (NZ), Andrej Brodnik (SLO), Andrew Paul Csizmadia (UK), Liesbeth De Mol (FR), Henry Hickman (NZ), Therese Keane (AU).

circostanza meriterebbe uno studio specifico, ma noi crediamo che il ritardo nell'instaurare un confronto e un dialogo approfonditi, volti a conciliare le sensibilità di orientamento pedagogico con quelle di orientamento scientifico, abbiano determinato una certa resistenza nel riconoscere il potenziale culturale ed educativo dell'informatica e, appunto, dell'attività di programmazione.

1. La confusione assai diffusa fra *uso* delle applicazioni informatiche e l'impresa *concettuale* di immaginarne l'utilità, progettarle, realizzarle, convalidarne il funzionamento e comprenderne l'impatto. Infatti, si è posta un'enfasi eccessiva sulla necessità di acquisire abilità di utilizzo di applicazioni specifiche, trascurando quasi completamente i principi scientifici che le hanno rese possibili. Un po' come se fosse sufficiente assaggiare molte meringhe con la panna per apprendere quanto la fisica, la chimica, o semplicemente la gastronomia siano cruciali in cucina: da questa enfasi mal riposta nemmeno i "pasticcieri" potrebbero trarre un beneficio significativo.
2. L'equivoco rapporto con discipline strutturate più stabilmente nella nostra tradizione scolastica, in particolare la matematica con la quale l'informatica ha spesso una relazione edipica (dando quasi l'impressione di volerla "superare" uccidendola), mentre sarebbe più proficuo se coltivasse le proprie potenzialità di nuovo strumento di pensiero, complementare e non alternativo ad altri più consolidati.
3. La percezione della programmazione come attività fortemente specialistica, utile solo a chi intende sviluppare una professionalità in settori tecnologici. Questa visione limitata trascura il fatto che l'elaborazione automatica di informazioni ha caratteristiche molto peculiari, ed è capace di influire grandemente sulla nostra consapevolezza del mondo, consapevolezza che è essa stessa basata su una elaborazione di informazioni — per definizione non automatica.

Le distorsioni a cui si è accennato sono emerse per motivi comprensibili, ma hanno avuto e tuttora hanno conseguenze negative. Per questo pensiamo possa essere utile mettere in luce, in una forma auspicabilmente accessibile a tutte le parti coinvolte in progetti educativi, gli aspetti centrali della programmazione, cioè *di che cosa parliamo* quando (noi informatici) sosteniamo che è importante fare l'esperienza della programmazione per capire alcuni aspetti del mondo che ci circonda senza subirne acriticamente l'impatto.

A tal fine abbiamo formato un gruppo di lavoro [5] che si è riunito a partire da marzo 2022 e che comprende studiosi con una lunga esperienza non solo nella didattica dell'informatica, ma anche nella storia e nella filosofia della disciplina, nonché nella diffusione delle tecnologie digitali. Ne è risultato un documento [6], che qui intendiamo riassumere, rivolto principalmente agli insegnanti e agli altri attori coinvolti nelle politiche scolastiche. L'obiettivo è fornire uno strumento che consenta di sviluppare una visione più chiara e articolata della *natura* dei programmi [1] e delle attività connesse alla loro realizzazione, e che inoltre possa fungere da guida ai fini della scelta dei temi più rilevanti per garantire l'efficacia delle azioni educative e pedagogiche.

2 Le sei "facce" di un programma

A prima vista potrebbe sembrare facile definire *che cos'è un programma*: in molti manuali, per esempio, è introdotto semplicemente come una sequenza di istruzioni per un *computer*. E in effetti termini come *coding*, molto usato anche nei documenti del Ministero dell'Istruzione, trasmettono di fatto questa idea riduttiva [7, 3]. Ma in realtà si tratta di un concetto molto più sfaccettato e ricco, come si può evincere da una lettura attenta della storia dell'informatica e da un'analisi di come l'elaborazione automatica di informazioni descritta dai programmi sia ormai inestricabilmente legata alla maggior parte delle nostre attività quotidiane, dalla guida delle automobili all'espressione

artistica. È ormai difficile, infatti, trovare ambiti rispetto ai quali la commistione con la programmazione non abbia influito in modi anche radicali. Attraverso le discussioni maturate in seno al gruppo di lavoro, alla luce delle prospettive maturate negli ambiti filosofico e didattico disciplinare, abbiamo identificato sei aspetti chiave che, a nostro avviso, sono i più salienti per caratterizzare la natura dei programmi e per spiegarne l'impatto rivoluzionario nella società. Non intendiamo certamente avanzare alcuna pretesa di esaustività: anche la visione d'insieme qui proposta è probabilmente riduttiva da certi punti di vista. Tuttavia, ci sembra sufficiente a motivare l'interesse che la Scuola dovrebbe dedicare all'argomento.

In sintesi, la caratterizzazione della natura dei programmi può essere così schematizzata:

1. I programmi sono *strumenti* utilizzabili;
2. I programmi sono artefatti tecnologici *opera dell'uomo*;
3. I programmi sono *oggetti fisici*;
4. I programmi sono *entità astratte*;
5. I programmi sono *eseguibili automaticamente*;
6. I programmi sono *artefatti linguistico-notazionali*.

Le sei "facce" così introdotte sono sintetizzate iconicamente nella figura 1, che inoltre presenta ai lati opposti di un esagono coppie di aspetti *duali*: essi caratterizzano la natura dei programmi proprio nella tensione dialettica che li contrappone.

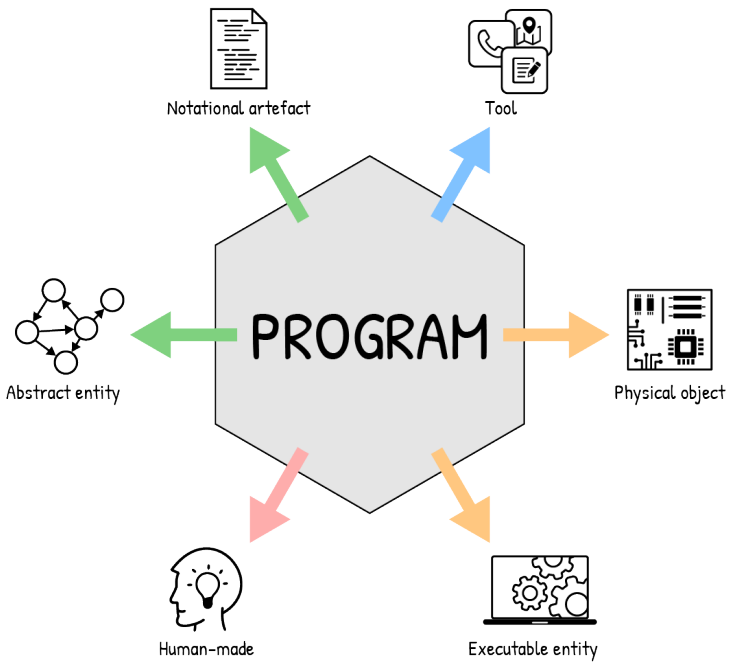

Figura 1: I sei aspetti chiave della natura di un programma
(figura tratta dal documento originale [6])

2.1 Programmi come strumenti

Questo è forse l'aspetto più evidente per molte persone. I programmi sono gli strumenti attraverso i quali svolgiamo moltissime delle nostre attività. In larga misura la società dipende dalla disponibilità di questi strumenti e i programmi ci permettono di affrontare compiti che altrimenti sarebbero al di fuori dalle nostre possibilità individuali, anche amplificando le nostre capacità cognitive. Grazie ai programmi lo stesso dispositivo fisico (p. es. uno *smartphone*) può assolvere molte funzioni diversissime fra loro: dalla contabilità aziendale, all'ascolto della musica, dalla fotografia al montaggio cinematografico, e così via. Tuttavia, è bene tenere presente che uno strumento non è mai neutrale, ma incorpora scelte e valori impliciti: occorre tener conto di questo anche quando la nostra esperienza è mediata da un programma — benché sia facile dimenticarsene a causa dell'ubiquità, della flessibilità e dell'invisibilità degli strumenti *software*.

2.2 Programmi come opera dell'uomo

I programmi sono un'opera umana, realizzati intenzionalmente per soddisfare qualche esigenza umana, con uno spettro molto ampio: si può scrivere un programma per risolvere un problema matematico, o portare a termine una ricerca in una banca dati, ma anche per soddisfare un impulso artistico o semplicemente per… imparare a programmare. In ogni caso si tratta di qualcosa costruito dall'uomo con uno scopo, che non sempre è allineato con gli scopi di chi il programma poi lo usa: chi sviluppa programmi che permettono la formazione di *social network* ha in genere obiettivi piuttosto diversi (p. es. raccogliere informazioni utili per il mercato pubblicitario) rispetto a chi usa tali programmi per condividere informazioni multimediali. La realizzazione di un programma richiede spesso un'attività complessa, svolta cooperativamente da gruppi di lavoro eterogenei che devono rispondere a moltissimi interessi e pulsioni, oltre a rispettare i vincoli tecnologici.

2.3 Programmi come oggetti fisici

I programmi sono anche oggetti fisici, un aspetto che è facile sottovalutare, ma che ha invece una grande rilevanza. Il programma che stiamo usando deve risiedere da qualche parte per poter essere eseguito da una macchina. È potenzialmente soggetto ad alterazioni, volontarie e involontarie, che possono impedirne il funzionamento oppure determinare effetti inaspettati/indesiderati. Inoltre, l'esecuzione di programmi e talvolta anche la semplice conservazione consumano energia.

2.4 Programmi come entità astratte

I programmi sono poi entità astratte: rappresentano manipolazioni di simboli, cioè di segni privi di significato intrinseco, cosicché il significato prende forma esclusivamente nella mente di chi li concepisce o ne prefigura il funzionamento. L'interprete (meccanico/elettronico), nel dare corso all'elaborazione descritta dal programma, non fa altro che operare in accordo con i suoi principi costruttivi, facendo evolvere lo stato dei suoi componenti in maniera prevedibile. L'interesse dell'elaborazione scaturisce dunque dalle proprietà astratte, algoritmiche dei programmi, che occorre imparare ad apprezzare e a mettere in relazione con la realtà concreta, valutandone le semplificazioni implicate e le potenziali alternative, oltre che la correttezza rispetto ai risultati attesi.

2.5 Programmi come entità eseguibili automaticamente

I programmi vengono eseguiti automaticamente, in contesti in cui non è previsto alcun intervento da parte dei progettisti, e ciò ne costituisce una caratteristica fondamentale. In altri termini, il programmatore deve immaginare ciò che accadrà o che potrebbe accadere durante l'esecuzione, ma una volta realizzato, l'esecuzione del programma è fuori dal suo controllo: per cambiare qualcosa

serve produrre una nuova versione. L'elaborazione dei dati di input (manuali o provenienti da dispositivi collegati e sensori) avviene nella maniera cieca e meccanica descritta al punto precedente; qualsiasi valutazione di ragionevolezza o affidabilità è responsabilità di coloro che hanno concorso allo sviluppo del sistema.

2.6 Programmi come artefatti linguistico-notazionali

I programmi sono inoltre uno strumento per comunicare idee e strategie non solo agli interpreti che li eseguono passivamente, ma anche ad altre persone che possono analizzarli, apprezzandone tutti i dettagli grazie alla precisione imposta dalla possibilità dell'esecuzione automatica. La varietà dei linguaggi di programmazione dimostra un'inesauribile ricerca di modalità espressive che si prestino alla modellazione e alla risoluzione dei problemi che si vogliono affrontare. Nello stesso tempo, le notazioni utilizzate per rappresentare l'informazione tendono intrinsecamente a condizionare il modo in cui riusciamo a ragionare per mezzo di esse.

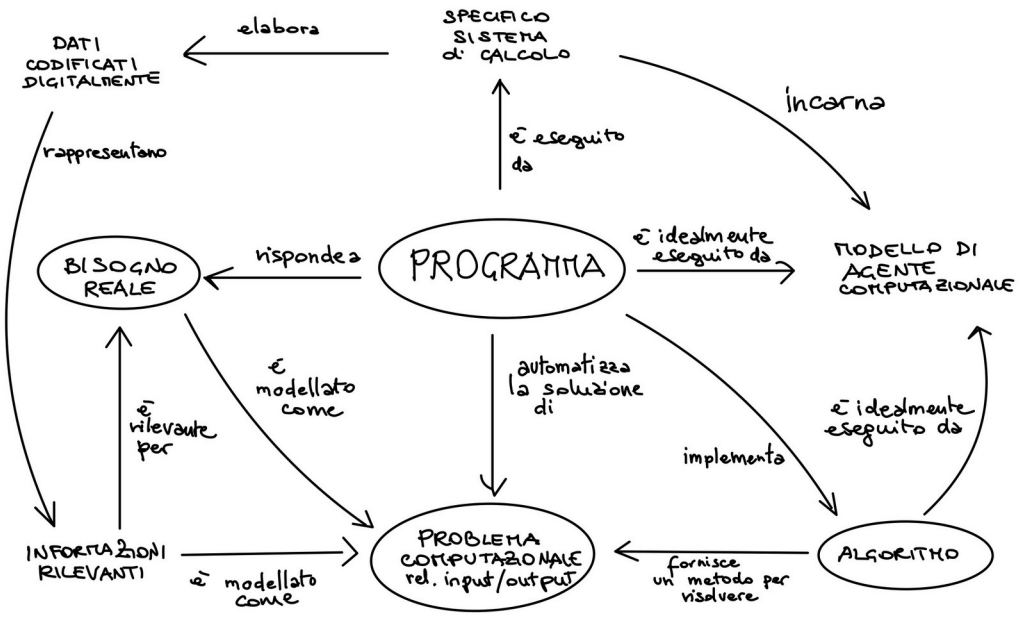

Figura 2: Una mappa concettuale che illustra l'attività della programmazione nel suo complesso (figura tradotta dal documento originale [6])

3 L'impresa concettuale delle elaborazioni automatiche

I programmi si scrivono per esprimere soluzioni automatiche di problemi computazionali e quindi assumono significato in relazione alle varie entità schematizzate nella mappa concettuale illustrata in figura 2. I problemi computazionali sono il corrispettivo in termini di elaborazione dell'informazione (ovvero, nel caso digitale, di manipolazione di rappresentazioni simboliche) di un'esigenza nel mondo

reale. I problemi computazionali modellano i bisogni nel mondo reale rappresentando le informazioni rilevanti con dati opportuni. Un programma può essere eseguito automaticamente — in altre parole un sistema fisico (per esempio un computer o uno smartphone) può dare corso all'elaborazione di informazioni descritta dal programma. Normalmente questi sistemi sono in grado di trattare solo rappresentazioni digitali, cioè simboliche, dei dati. I dati, perciò, devono essere codificati tramite sequenze di simboli tratti da un alfabeto predefinito.

Un algoritmo è l'idea astratta di un metodo (ossia una procedura precisa che può essere portata a termine seguendola in modo *prescrittivo*) per risolvere un problema computazionale. L'algoritmo viene progettato pensandone l'esecuzione da parte di un agente computazionale idealizzato, capace di eseguire un insieme ridotto di azioni primitive. Un sistema di calcolo reale (potrebbe essere un computer o un altro dispositivo hardware, ma anche un altro programma, come l'interprete di un linguaggio di programmazione) è un esemplare specifico di tale agente computazionale idealizzato. Affinché possa essere eseguito automaticamente da un sistema di calcolo reale, un algoritmo deve essere implementato, cioè codificato in un linguaggio di programmazione rispettandone le rigide regole necessarie a evitare ambiguità e adattandolo ai vincoli tecnologici specifici del sistema: solo così un algoritmo si traduce effettivamente in un programma.

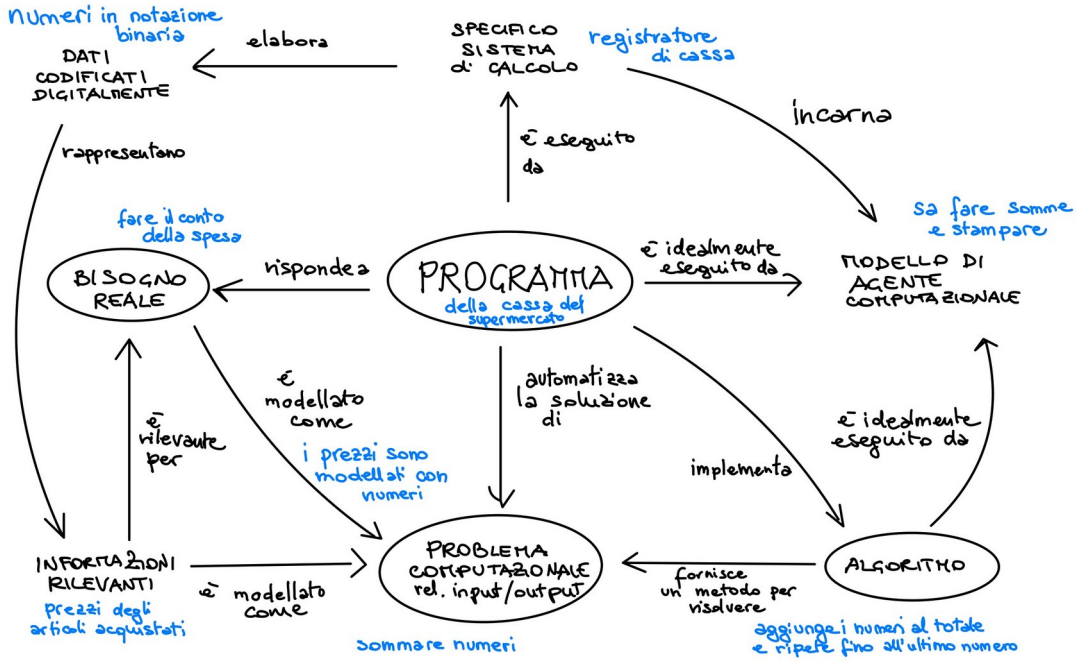

Figura 3: Un esempio: il programma della cassa del supermercato
(figura tradotta dal documento originale [6])

3.1 Un esempio

La figura 3 illustra i concetti che appaiono nella mappa di figura 2 facendo riferimento a un programma (quello della "cassa del supermercato") progettato per gestire la produzione degli scontrini di un negozio. I prezzi degli articoli venduti possono essere modellati come numeri. Perciò,

l'elaborazione di uno scontrino corrisponde al generico problema computazionale di "sommare una serie di numeri". Il programma della cassa del supermercato è eseguito da un registratore di cassa, un dispositivo in grado di acquisire i prezzi degli articoli in vendita (digitati da un operatore o letti interpretando un codice a barre), quindi di stampare gli scontrini corrispondenti. Il dispositivo, in realtà, manipola i numeri rappresentati tramite sequenze di simboli binari — basati cioè su due stati chiaramente distinguibili dei suoi componenti elettronici.

L'algoritmo tipico per sommare una serie di numeri è il seguente:

1. usa una variabile per tener traccia del totale, assegnandovi inizialmente il valore zero;
2. fintantoché la lista dei numeri non è esaurita, ripeti l'istruzione seguente:
   2.1. somma il prossimo numero alla variabile che tiene traccia del totale;
3. stampa il valore della variabile che tiene traccia del totale e termina.

Durante la progettazione di questo algoritmo, assumiamo che l'esecutore sia in grado di ricevere in input qualsiasi numero, sommare due numeri e stampare qualsiasi numero. Affinché il registratore di cassa possa eseguire *fisicamente* queste operazioni occorre tener conto di alcuni vincoli specifici, per esempio i limiti ai valori ed eventualmente alla quantità di numeri oggetto dell'elaborazione. L'algoritmo probabilmente viene implementato in un linguaggio di programmazione specializzato per i registratori di cassa. Sia il problema computazionale che l'algoritmo, però, sono molto generali e possono essere facilmente adattati ad altri scopi analoghi, come per esempio calcolare la popolazione totale di un Paese sulla base delle popolazioni delle regioni che lo costituiscono. Perciò, un'implementazione in un linguaggio di programmazione non specifico permetterebbe di usare lo stesso programma, magari con qualche piccolo aggiustamento, anche in contesti piuttosto diversi.

4 Conclusione

Lo scopo del gruppo di lavoro è fornire una chiara esposizione degli aspetti chiave dei programmi che possa aiutare gli insegnanti e altri operatori culturali a identificare le ragioni della centralità del *software* nella società attuale e a orientare al meglio gli interventi educativi. La versione italiana preliminare del documento è disponibile all'indirizzo https://aladdin.unimi.it/naturadeiprogrammi.pdf. La versione definitiva è attesa per la fine del 2022 e crediamo che possa diventare una risorsa importante ai fini di progettare percorsi formativi in cui la pratica della programmazione e la riflessione sui programmi vengano intesi innanzitutto come strumenti di consapevolezza e cittadinanza attiva.

Riferimenti bibliografici

[1] Andrej Brodnik, Andrew Csizmadia, Gerald Futschek, Lidija Kralj, Violetta Lonati, Peter Micheuz, and Mattia Monga. Programming for all: Understanding the nature of programs. *CoRR*, abs/2111.04887, 2021.

[2] Caena, Francesca & Redecker, Christine. (2019). Aligning teacher competence frameworks to 21st century challenges: The case for the European Digital Competence Framework for Educators (*Digcompedu*). European Journal of Education. 54. 10.1111/ejed.12345.

[3] Isabella Corradini, Michael Lodi, and Enrico Nardelli. An investigation of italian primary school teachers' view on coding and programming. In Sergei N. Pozdniakov and Valentina Dagienė, editors, *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, pages 228–243, Cham, 2018. Springer International Publishing.

[4] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. An international comparison of k-12

computer science education intended and enacted curricula. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, Koli Calling '19, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Violetta Lonati, Andrej Brodnik, Tim Bell, Andrew Paul Csizmadia, Liesbeth De Mol, Henry Hickman, Therese Keane, Claudio Mirolo, Mattia Monga, and Matti Tedre. Characterizing the nature of programs for educational purposes. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2*, ITiCSE '22, page 572–573, New York, NY, USA, 2022. Association for Computing Machinery.

[6] Violetta Lonati, Andrej Brodnik, Tim Bell, Andrew Paul Csizmadia, Liesbeth De Mol, Henry Hickman, Therese Keane, Claudio Mirolo, and Mattia Monga. The nature of programs, or: What we talk about when we talk about programs. https://drive.google.com/file/d/1lhVPDhSHu3ivRXvcK7BVEhSmpqz5izow (ultimo accesso, novembre 2022).

[7] Violetta Lonati, Dario Malchiodi, Mattia Monga, and Anna Morpurgo. Is coding the way to go? In Andrej Brodnik and Jan Vahrenhold, editors, *8th International Conference on Informatics in Schools: Situation, Evolution, and Perspective*, volume 9378 of *LNCS*, pages 165–174, Switzerland, September 2015. Springer International Publishing.