Indice

APPENDICE

Introduzione

Il percorso proposto in questo testo è il frutto di un lungo processo di ricerca didattica iniziato trent'anni fa quando come docente di Laboratorio di Matematica e di Laboratorio di Calcolo delle Probabilità, Statistica e Ricerca Operativa, ovvero di Informatica applicata a metodi specifici di tali discipline, insegnavo nel triennio della specializzazione Informatica in quelli che erano allora gli Istituti Tecnici Industriali Statali.

Nel mio percorso scolastico ero stata avviata allo studio della Matematica e dell'Informatica, quelle bellissime Scienze che insegnano ad apprezzare l'eleganza nella dimostrazione di un Teorema o nella composizione del codice che risolve un problema particolarmente complesso. Il lavoro invece mi richiedeva di approfondire gli aspetti applicativi di quelle due discipline: davvero un bel connubio!

Tutto ciò mi spingeva dunque a porre particolare attenzione su quanto la nostra vita fosse pervasa di Matematica (già da piccolini inconsapevolmente la applichiamo quando dividiamo le caramelle con gli amici), ma anche di Matematica e Informatica insieme: nell'inserire le nostre credenziali per accedere ad una applicazione, o nell'inviare un messaggio in una chat, entra in gioco la crittografia che si basa sui numeri primi! Si tratta di applicazioni Informatiche della Matematica o di applicazioni Matematiche dell'Informatica?

Nel mio lavoro poi, proponendo, di realizzare applicazioni Informatiche/Matematiche ai miei studenti, troppo spesso capitava di vederli ansiosi di sedersi davanti alla tastiera per "pigiare tasti"! Purtroppo, se non ci si sofferma almeno a

comprendere il problema posto, l'azione che si esegue al pc non è certo quella del programmatore, bensì, come scherzosamente amavo dire, del "pigiatore di tasti". Di conseguenza, la maggior parte dei programmi prodotti con quell'atteggiamento risultavano contorti, poco efficienti e difficili da correggere in caso di errori. Si trattava evidentemente di studenti poco propensi a dedicarsi a quell'attività nota come "analisi dei problemi".

Nel 2009 la Riforma scolastica ha introdotto la disciplina Tecnologie Informatiche nelle classi prime dei nuovi Istituti Tecnici Tecnologici, la cui parte laboratoriale poteva essere insegnata dalla mia classe di concorso, e ha contestualmente eliminato le due discipline alle quali mi ero precedentemente dedicata.

Quella che inizialmente mi era sembrata una catastrofe, una vera costrizione, si è rivelata invece una grande opportunità offrendomi la possibilità di approfondire quell'aspetto che mi aveva da sempre affascinata: il pensiero algoritmico e le tecniche per il suo sviluppo. Si tratta del processo cognitivo esplorato da Papert[1] nelle sue ricerche al MIT e raccontato magnificamente nel suo Mindstorm, una pietra miliare per la mia formazione di informatica, cioè quello stesso percorso creativo che attua il programmatore prima dell'azione di codifica mediante linguaggi di programmazione strutturati.

Ma se Papert ha dichiarato:

> *"When you learn to program a computer you almost never get it right the first time"*[2]

è anche vero che l'analisi di un problema effettuata in modo frettoloso, o addirittura inesistente come nel caso dei miei stu-

---

[1] http://www.papert.org/

[2] "Quando impari a programmare un computer non riesci quasi mai a farlo bene la prima volta".

denti, porta a programmare un computer non così bene anche al secondo o al terzo tentativo!

Papert ha anche posto l'attenzione sul giusto atteggiamento rispetto all'errore, il *bug*, che deve essere inteso come momento di crescita, e sul processo di ricerca e soluzione dell'errore, il *debugging*, importante tanto quanto l'esercizio della programmazione vera e propria. Ma, per la mia esperienza, si può evitare che *bug* e *debugging* possano essere vissute come esperienze davvero traumatiche semplicemente affrontando con maggiore consapevolezza l'analisi del problema in questione.

Perché il titolo del testo si riferisce al pensiero algoritmico e non a quello computazionale? Negli ultimi anni, da quando cioè è stato "scoperto" che esisteva un modo di fare programmazione alla portata di tutti, anche dei più piccoli, denominato *coding*, sono fiorite le discussioni sul "pensiero computazionale", locuzione coniata da Jeannette Wing in un famoso articolo del 2006[3] che ha scatenato interessanti discussioni: esiste o non esiste? ha senso che metodi tipici della programmazione vengano freddamente applicati anche nella vita quotidiana? Molti hanno sottolineato, più filosoficamente parlando, che la vita è fatta anche di sentimento ed emozione, non solo di "computazione".

In questo testo non ho alcuna intenzione di addentrarmi in tale discussione e quindi ho preferito riferirmi a questo modo di "pensare" con la locuzione "pensiero algoritmico", pur trovando nell'articolo della Wing molte affinità col mio pensiero, ma soprattutto con la mia esperienza di persona che per formazione e per lavoro ha fatto propri i metodi della programmazione. Infatti, mi sono ritrovata nel tempo, piano piano, ad applicarli, quasi sempre senza premeditazione, anche nella vita quotidiana ogniqualvolta vi fosse la necessità di organiz-

---

[3] Articolo originale in cui Jeannette Wing ha coniato la locuzione "pensiero computazionale" http://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

zare e progettare; per questo la mia convinzione è la stessa della Wing quando scrive "*Ideas, not artifacts ...For everyone, everywhere ...*", cioè che sia possibile per chiunque ripetere la mia stessa esperienza.

Spero anche di riuscire a mostrare come sia preferibile affrontare una qualsiasi situazione con consapevolezza, in modo da poter scegliere la propria via per risolverla, e come la descrizione di un pensiero complesso, se realizzata con gli strumenti del pensiero algoritmico, si presenti inaspettatamente semplice ed elegante, tanto da poter essere ammirata alla stessa stregua di un'opera d'arte.

Gli aspetti fondamentali che mi hanno spinta verso la realizzazione del percorso didattico proposto in questo testo sono:

- la constatazione che prima o poi nella vita ad ognuno possa capitare di dover "fare progetti"; questo mi ha suggerito che i metodi didattici utilizzati nella formazione di un buon tecnico progettista (obiettivo finale del triennio di specializzazione degli ITTS, indipendentemente dalla specializzazione scelta) possano essere applicati da tutti, seppure declinati con livelli diversi di approfondimento
- la verifica sul campo che un approccio metodologico e strutturato sia applicabile a problemi di qualunque natura; quello che eventualmente può cambiare, a seconda dei casi, è la scelta degli obiettivi finali che da generali e quindi perseguibili anche nella vita quotidiana (si pensi ad una ricetta culinaria), possono trasformarsi in obiettivi specifici (codificare algoritmi mediante linguaggi di programmazione)
- la consapevolezza che spesso non si possa disporre di adeguati strumenti tecnologici; da qui la necessità di sviluppare un percorso attuabile anche in modalità *unplugged*, ma che al tempo stesso possa fornire gli strumenti di base tipicamente utilizzati in ambito informatico non solo per un eventuale sviluppo futuro di applicazioni, ma anche per perseguire l'obiettivo del "*Thinking about*

thinking", ricordando le parole di Papert, che diventa più stimolante quando si prospetta un risvolto informatico al processo risolutivo

→ la constatazione che conoscere le più svariate tecnologie, sia software che hardware, seppure a livelli di base, possa consentire di effettuare scelte consapevoli nel loro uso e quindi contribuisca all'acquisizione della competenza di cittadinanza digitale; questo mi ha suggerito di proporre strumenti diversificati nella fase applicativa finale di ogni situazione problematica proposta nel testo.

Ritenevo poi importante realizzare un percorso da affiancare a quello del libro di testo in adozione per le mie classi, in modo che gli alunni potessero sviluppare anche le competenze relative all'utilizzo di informazioni provenienti da più fonti e al loro confronto.

Perché nel sottotitolo mi riferisco alle "Le sfide di Marco e Sofia"?

Nel tempo, prendendo spunto da situazioni della quotidianità, ho costruito una collezione di problemi al fine di proporre ai miei studenti esercitazioni di laboratorio sempre nuove e diversificate. Ho iniziato quindi a fingere che realmente esistesse un ragazzo di nome Marco, o a volte si trattava di una ragazza, Sofia, che mi avesse sottoposto quei problemi, chiedendo aiuto nella loro risoluzione. Questa finzione, colta comunque fin da subito dai miei studenti, creava un clima di complicità che ci permetteva anche con leggerezza di scherzare sul fatto che a Marco e a Sofia ne capitasse sempre una di nuova. Dopo poche lezioni la frase "Poveri Marco e Sofia, ne hanno sempre una!" era diventata un tormentone, che ai ragazzi piace molto, ed era venuto a crearsi un clima ideale per veicolare concetti importanti e non sempre di facile comprensione, ma soprattutto per far sì che tali concetti si consolidassero nella memoria dei ragazzi.

In questo libro Marco e Sofia sono diventati due studenti della classe terza di una scuola secondaria di primo grado molto speciale (si veda lo *storytelling* in appendice, disponibile anche nei materiali digitali del testo).

Le situazioni proposte sono state studiate in modo da affrontare aspetti che gradualmente possano rendere più complessi i processi logico-cognitivi e permettano al "ricercatore della soluzione" di osservarle da punti di vista sempre diversi. Sono presentate come Sfide, data la connotazione decisamente più positiva di questo termine rispetto a problema, che un gruppo di alunni si trovano ad affrontare nella loro pratica quotidiana in una scuola speciale, aperta ad una didattica innovativa nei metodi (pochi momenti di lezione frontale e prevalenza di didattica per progetti) e negli strumenti tecnologici utilizzati. Gli alunni affrontano tali Sfide non tanto perchè uno dei docenti le abbia loro assegnate, ma perché l'atteggiamento di curiosità, proprio di ciascuno dei protagonisti, li induce ad approfondire la ricerca per una crescita personale. Si tratta, in buona sintesi, degli alunni ideali che ogni docente vorrebbe avere!!!

Ogni situazione diventa pretesto per applicare un concetto algebrico, geometrico o statistico che costituisce lo strumento teorico indispensabile per individuare la strategia risolutiva di quella specifica questione. La conseguente descrizione dell'algoritmo può essere intesa come atto conclusivo, ma già di grande valenza cognitiva, del progetto didattico proposto, oppure come passo necessario per la successiva attività di *coding*. Inoltre, i suggerimenti affinché ogni situazione possa essere ampliata e resa più complessa, a discrezione del "conduttore" dell'attività, rappresentano un percorso atto a sviluppare nel "ricercatore" la consapevolezza che quel particolare caso appartenga in realtà ad una categoria di casi e che la strategia risolutiva individuata possa essere generalizzata per risolvere qualsiasi situazione di quella categoria.

Per quanto concerne la metodologia didattica, l'allenamento allo sviluppo del pensiero algoritmico viene realizzato attuando le seguenti fasi:

- ➞ presentazione della situazione problematica attraverso lo strumento dello *storytelling*
- ➞ analisi della situazione: piena comprensione dell'obiettivo da raggiungere e ricerca dei dati noti, o loro scelta qualora il problema posto sia aperto al punto tale da consentire di formulare più ipotesi iniziali
- ➞ ricerca di una o più strategie risolutive, mettendo in atto metodi e strumenti specifici di quel preciso ambito, non per "trovare il risultato", bensì per individuare il processo risolutivo che porta al risultato; è importante comprendere le motivazioni più profonde per le quali vale la pena affrontare questo sforzo, spiegate magnificamente da Papert:

  "And in teaching the computer how to think, children embark on an exploration about how they themeselves think"[4]

- ➞ scelta della strategia risolutiva migliore, secondo determinati parametri fissati a priori, ad esempio in relazione al tempo di elaborazione, all'uso di risorse, all'originalità e all'eleganza della soluzione, e descrizione della strategia scelta mediante opportuno linguaggio di progetto
- ➞ eventuale scelta di un ambiente di sviluppo per implementare un'applicazione informatica.

È importante anche ricordare che le osservazioni e gli approfondimenti sono stati studiati in modo da consentire al lettore-docente la costruzione di un proprio percorso didattico:

---

[4] "E nell'insegnare al computer come pensare, i bambini intraprendono un'esplorazione del loro stesso pensiero".

- può scegliere una Sfida e seguire con gli alunni le indicazioni proposte nel testo per la sua risoluzione;
- può leggere lo *storytelling* insieme agli alunni o, in applicazione della metodologia della *flipped classroom*, lasciare a loro il compito di leggere e individuare quali siano le situazioni problematiche che richiedano una soluzione algoritmica;
- può fermarsi alla produzione dell'algoritmo risolutivo dopo la fase di analisi di una situazione iniziale, oppure andare oltre suggerendo possibili generalizzazioni della questione oggetto di analisi, anche guidando gli studenti affinché siano essi stessi a proporle;
- e ancora, può fermarsi alla fase algoritmica, oppure proseguire con la scelta di uno strumento software per realizzare un prodotto informatico;
- infine, può proporre di applicare lo strumento software di sviluppo suggerito in una Sfida anche per implementare applicazioni relative ad altre Sfide.

È evidente quindi che questo testo non intende occuparsi esclusivamente di *problem solving.* Infatti, il processo messo in atto in ambito informatico non appena viene affrontata l'analisi di un problema, va decisamente oltre il *problem solving*: non ci si accontenta di trovare la soluzione, ma ci si impegna a descriverla in modo che possa naturalmente evolvere in applicazione informatica. Si tratta quindi di un processo che, in linea con le osservazioni della Wing, allena la mente ad individuare percorsi risolutivi strutturati che, come già ampiamente descritto, consentano anche di mettere le basi per realizzare con poco sforzo la fase finale di *coding*.

Per quanto riguarda quegli aspetti più strettamente algoritmici, la rappresentazione dei dati mediante l'uso di *variabili* viene affrontata in modo graduale, proponendo dapprima situa-

zioni con dati costanti; compito del "conduttore" sarà quindi indurre il "ricercatore" nell'individuare in seguito la soluzione del problema generalizzato, sostituendo i dati costanti con quelli variabili. Non mancano poi occasioni per accompagnare il lettore verso l'uso di strutture dati tipiche dei linguaggi ad alto livello come ad esempio le *liste* e i *dizionari*.

Tutte le situazioni hanno soluzione che può essere implementata con la struttura *sequenza* (cioè con una serie di azioni che devono essere eseguite una dopo l'altra – si tratta della struttura computazionale più semplice); gli approfondimenti proposti conducono ad un'apertura verso le altre strutture computazionali: la *selezione* e i *cicli*. Inoltre, dove possibile e conveniente, vengono introdotti i concetti di *procedura e funzione*.

Come già detto, per tutte le attività vengono fornite diverse chiavi di lettura: possono essere realizzate interamente *unplugged* limitandosi alla descrizione degli algoritmi mediante un linguaggio di pseudo-codifica, oppure si può proseguire con la produzione di applicazioni di vario tipo. Nella proposta di strumenti software si va dai linguaggi visuali a blocchi come Scratch, MakeCode, Blockly e AppInventor, più adatti ad alunni della scuola secondaria di primo grado che iniziano il loro approccio con la programmazione, fino ai linguaggi testuali ad alto livello Python e C; vengono proposte applicazioni realizzate con il foglio di calcolo nelle sue varie declinazioni d'uso e l'utilizzo di ambienti di *coding* e repository come Trinket, senza dimenticare strumenti come la scheda di prototipazione micro:bit e il single-board computer Raspberry.

In appendice, oltre allo *storytelling* già citato, si trova anche "Un esercizio da programmatori". Si tratta di uno dei classici esercizi che vengono proposti a chi inizia un corso di studio della programmazione, ma affrontato col metodo proposto in questo testo. La cosa curiosa è che il più delle volte vengono proposti agli studenti di programmazione dei problemi che trovano difficilmente applicazione nella pratica quotidiana e, alla

loro obiezione “A cosa mi serve?”, la risposta non può che essere “Ad aprire la mente” … davvero un ottimo obiettivo! Ma in questo testo si è voluto mettere in campo un approccio diametralmente opposto: partire dalla quotidianità per individuare situazioni che possano richiedere soluzione algoritmica, da trasformare eventualmente in applicazione informatica. Ritengo utile però mostrare come il metodo suggerito in questo testo possa essere pervasivo, quindi applicabile a più contesti.

Nel dialogo e nel confronto con i miei colleghi delle materie di specializzazione dell’ITTS, ho avuto conferma che gli studenti che avevano seguito il percorso, così come proposto in questo testo, nel loro primo anno di biennio, riuscivano poi più facilmente ad affrontare con adeguati strumenti quelle situazioni che tradizionalmente si presentano durante la realizzazione di progetti laboratoriali nel secondo biennio e nell’ultimo anno di specializzazione.

L’incontro con il collega Gennaro Nasti, docente di Tecnologia nella scuola secondaria di primo grado, è poi stato illuminante: sua è stata l’idea di ambientare le Sfide in una specialissima scuola media, idea da me prontamente colta. Il confronto circa le nostre differenti esperienze didattiche, ci ha indotti a constatare che il percorso da me ideato potesse essere attuabile anche nelle classi della scuola secondaria di primo grado, ovviamente con opportune scelte e semplificazioni individuate di volta in volta dai singoli docenti, proprio per la sua caratteristica di offrire diversi livelli di lettura e di applicazione; inoltre, abbiamo convenuto che i percorsi proposti nel testo avrebbero potuto trovare attuazione nei progetti di continuità tra scuola secondaria di primo grado e secondaria di secondo grado.

Questo testo si rivolge quindi ai docenti di entrambi i gradi di scuola secondaria che ritengano indispensabile allenare i propri studenti nell’applicazione del pensiero algoritmico, avvicinandoli alle applicazioni software e hardware solo dopo aver

consolidato le competenze relative all'analisi di problemi. Ma è anche rivolto ai genitori e ai tanti volontari che al di fuori del mondo scolastico si attivano per avvicinare bambin* e ragazz* al *coding*, ad esempio nei *coderdojo*, e che desiderino utilizzare una metodologia per offrire esperienze più consapevoli.

Come già accennato, non vuole essere però un manuale tecnico informatico, bensì proporre diversi ambienti di sviluppo software e piattaforme hardware per mostrare a bambin* e ragazz* le enormi potenzialità messe a disposizione dalla Scienza Informatica come supporto nello sviluppo della creatività individuale.

Preciso, inoltre, che successivamente alla stesura del testo alcuni degli ambienti software proposti sono stati aggiornati con nuove versioni e questo evento continuerà ad accadere a seguito della velocità di sviluppo della tecnologia; ciò non rende obsolete le applicazione suggerite, le quali continuano ad essere un valido punto di partenza per gli approfondimenti che ogni docente vorrà realizzare con i propri studenti.