Un curricolo per separare l'informatica dall'applimatica*
in un Liceo scientifico opzione scienze applicate

Gionata Massi

IIS Savoia Benincasa, Ancona (AN)

Abstract

Questo contributo presenta un percorso pensato per l'insegnamento dell'informatica nel Liceo scientifico opzione scienze applicate, andando a concepire la materia come una disciplina scientifico/ingegneristica[1] che coniuga creatività e metodo.

Il curricolo si concentra sui traguardi di competenza fissati dalle Indicazioni nazionali [1] ma presenta una selezione di argomenti e contenuti che sconcertano studenti, genitori e colleghi.

Le aspettative sembrano concentrarsi su due grandi stereotipi, da una parte quello dell'applimatica*, ossia del mero uso del computer e delle tecnologie del momento, dall'altra quella dell'apprendimento di un linguaggio di programmazione, in genere uno e uno solo, frequentemente il C++[2]. I libri di testo seguono questi stereotipi o aggiungono quello del *coding*.

La sperimentazione ha riguardato due classi e si è conclusa per l'esigenza di uniformare l'offerta didattica tra classi parallele con insegnanti diversi che, in mancanza di validi supporti didattici trovano estremamente oneroso sviluppare ex novo l'attività didattica.

In conclusione si considera la proposta della didattica dell'informatica come formativa e orientante ma troppo complessa in assenza di Open Educational Resource (OER) in lingua italiana e di qualità.

Contents

*neologismo trovato in alcuni lucidi del prof. Mattia Monga

[1] Harold Abelson e Gerald Jay Sussman, gli autori di SICP potrebbero non essere d'accordo sulla natura dell'informatica

[2] Bjarne Stroustrup è contento ma anche lui sembra consapevole del fatto che il linguaggio non tenga in minimo conto le esigenze didattiche.

1 Introduzione

Gli insegnanti di informatica italiani, a differenza dei colleghi di molte altre nazioni, non hanno sillabi, programmi nazionali e sistemi di valutazione sviluppati per l'insegnamento della loro materia ma devono programmare l'attività didattica mediando fra vincoli normativi e disparate esigenze.

Negli Stati Uniti ci sono curricola e attività didattiche progettate in dettaglio, come [13] che propone scopi affini alle indicazioni nazionali [1]. Le esperienze e i curricula sono spesso raccolti in catalogati come [4]. Per i curriculum come lo Advanced Placement Computer Science [10] sono offerti vari corsi gratuiti anche da piattaforme come [3] e [5]. La comunità scientifica è attiva nelle proposte didattiche e sviluppa linguaggi e sistemi ad hoc[3] ([7] e [8], [15]).

In Nuova Zelanda hanno sviluppato la *Computer Science Field Guide* [11], una guida interattiva che così interessante da essere tradotta in varie lingue ma non in italiano.

In Gran Bretagna c'è molta attenzione all'insegnamento dell'informatica e alla formazione professionale dei docenti[4] [14].

In Italia ogni insegnante, attenendosi alle Indicazioni nazionali e al PTOF della scuola in cui insegna, considerate le necessità del territorio, tenuto conto delle aspettative di studenti, dei genitori e dei colleghi e di eventuali libri testo, scrive la sua programmazione didattica.

Si propone una di queste programmazioni costruita rilassando i vincoli sulle aspettative e sui testi, prevendendo attività alla lavagna e, se al PC, usando esclusivamente Software Libero e formati aperti.

2 Primo biennio

Si propongono alcune attività del primo biennio in sintonia con le indicazioni nazionali.

2.1 Scrittura tecnico-scientifica al calcolatore

Per soddisfare il requisito per cui 'lo studente conosce gli elementi costitutivi di un documento elettronico e i principali strumenti di produzione' e la pressante richiesta dei colleghi sull'uso di un sistema di videoscrittura[5], si è proposta un'unità didattica di introduzione alla comunicazione tecnico/scientifica per il conseguimento delle seguenti abilità:

1. Identificare gli attori e di una comunicazione
2. Distinguere tra dati, informazioni e conoscenza
3. Aggiungere l'autore e il titolo nel frontespizio di una relazione
4. Suddividere un testo nei suoi elementi strutturali ed aggiungere i titoletti
5. Scrivere tabelle, immagini e altri contenuti flottanti con didascalie

[3] Seymour Papert e Mitchel Resnick sono così famosi tra gli insegnati italiani che è inutile citarli.

[4] Un progetto di riferimento è [6] ma tante comunità e fondazioni di produttori di hardware Open Source che propongono curricula completi.

[5] Per l'insegnate tipo il programma di videoscrittura è solo quella versione del noto produttore di sistemi operativi di cui egli ha ottenuto una qualche licenza

6. Scrivere con elenchi ordinati, non ordinati e descrittivi
7. Enfatizzare il testo selezionando un carattere tipografico opportuno

Si è scelto AsciiDoctor[2] con LaTeXe `asciidoctor-diagram` come caso di studio. L'autore ha costruito un'applicazione web basata per la sperimentazione e la verifica automatica del livello di apprendimento. La didattica del linguaggio è stata condotta con esempi e generalizzazioni della sintassi tramite diagrammi rail-road.

Come risultato qualche studente ha usato il sistema per produrre relazioni delle esperienze di fisica scritte e composte tipograficamente in modo impeccabile.

2.2 Dati e codifiche

Per soddisfare il requisito sulla '... una introduzione alla codifica binaria ... i codici ASCII e Unicode', si è focalizzato su:

1. Convertire un numero da base due a base dieci
2. Convertire un numero da base dieci a base due
3. Codificare un testo usando codifiche binarie dei caratteri
4. Decodificare un testo codificato in binario
5. Comprendere le esigenze che hanno condotto allo sviluppo dello standard UNICODE
6. Codificare un'immagine monocromatica mediante run-length encoding
7. Decodificare un'immagine monocromatica codificata mediante run-length encoding

Lo strumento didattico di riferimento è stato il testo 'Computer Science Unplugged' [9].

L'apprendimento delle codifiche è stato reso più attraente rispetto alle modalità didattiche classiche.

2.3 Problemi, modelli, soluzioni e algoritmi

Per introdurre gli studenti alla programmazione vengono prima presentati e formalizzati i problemi. Si sono scelte le seguenti abilità:

1. Saper formalizzare un problema di ricerca
2. Simulare l'esecuzione dell'algoritmo di ricerca lineare
3. Simulare l'esecuzione dell'algoritmo di ricerca binaria
4. Saper formalizzare il concetto di ordinamento di una sequenza
5. Simulare l'algoritmo di ordinamento per selezione, per inserimento e a bolle
6. Calcolare il numero di confronti e di scambi degli algoritmi di ordinamento basati su confronti e scambi
7. Comprendere i criteri di scelta di un algoritmo rispetto ad altri

8. Astrarre il modello di semplici problemi di natura quantitativa e descrivere algoritmicamente il procedimento di soluzione

9. Simulare l'esecuzione di un programma

Anche in questo caso si è fatto ricorso a 'Computer Science Unplugged'.

2.4 Interagire col sistema operativo tramite la shell

Per introdurre il sistema operativo si è scelto di saper interagire con il suo livello più esterno sviluppando le seguenti abilità:

1. Creare, rinominare, spostare e cancellare un file

2. Organizzare i file nelle directory

3. Invocare un programma

4. Elencare file e directory

5. Creare una pipe anonima tra due o più utility

6. Usare comandi per la data, l'estrazione della testa e della coda di un file di testo, l'ordinamento delle linee, l'estrazione di campi

Si è fatto ricorso ad una connessione 'ssh' ad un server GNU/Linux.

3 Secondo biennio e quinto anno

3.1 Documenti elettronici per il web

Nel secondo biennio si è cercato di sviluppare, gradualmente, l'astrazione e di introdurre vari paradigmi di programmazioni, tranne quello detto *Banana Gorilla Jungle* perché con due ore a settimana bisogna fare delle scelte.

3.2 Fogli elettronici

I fogli elettronici sono richiesti dai colleghi e si possono introdurre i concetti di reattività ma anche di funzione di ordine superiore in modo intuitivo. Si sono stabiliti i seguenti traguardi:

1. Usare le operazioni di filtraggio, riduzione e mappa nel foglio di calcolo

2. Usare il foglio di calcolo per modellizzare e risolvere le problematiche d'interesse per il corso di studi

3. Usare il risolutore per problemi di scelta

4. Impostare problemi a variabili intere con il risolutore

3.3 Paradigmi di programmazione

Nell'incapacità dell'autore di decodificare le richieste di 'implementazione di un linguaggio di programmazione, metodologie di programmazione, sintassi di un linguaggio orientato agli oggetti', si è scelto di presentare il paradigma della programmazione strutturata (con funzioni definite nell'ambito di una funzione) in JavaScript, quello logico con esempi in Prolog, e quello funzionale in JavaScript, realizzando funzioni pure espresse senza l'uso dei costrutti `while`, `do-while` e textttfor.

1. Saper realizzare algoritmi usando le funzioni e i costrutti di sequenza, selezione e iterazione (paradigma strutturato)
2. Progettare semplici basi di fatti e regole con paradigma logico
3. Saper realizzare semplici algoritmi mediante funzioni sintatticamente ricorsive

3.4 Problemi su grafi e combinatorici

1. Saper rappresentare un grafo
2. Realizzare gli algoritmi di visita di un albero binario: preordine, postordine, simmetrico
3. Visitare un grafo in profondità
4. Visitare un grafo in ampiezza
5. Simulare l'algoritmo di Dijkstra
6. Rappresentare sul foglio di calcolo il problema dello zaino

3.5 Funzioni di ordine superiore

1. Definire funzioni che hanno come argomento una funzione di variabile reale e un valore reale e che restituiscono un valore reale
2. Definire funzioni che hanno come argomento una funzione di variabile reale e che restituiscono una funzione di variabile reale
3. Descrivere gli algoritmi in termini di applicazione e composizione di funzioni
4. Manipolare gli array usando le tecniche: filter, map, reduce.

3.6 Radici di funzioni

1. Saper applicare il metodo di Erone di Alessandria per il calcolo delle radici quadrate
2. Riconoscere l'esistenza di una radice di una funzione dato un intervallo
3. Saper effettuare la ricerca numerica di una radice tramite il metodo di bisezione
4. Realizzare in un linguaggio di programmazione e al foglio di calcolo il metodo di bisezione
5. Saper effettuare la ricerca numerica di una radice tramite il metodo delle tangenti (di Newton)

6. Realizzare in un linguaggio di programmazione e al foglio di calcolo il metodo delle tangenti (di Newton)
7. Saper riconoscere l'equivalenza di due metodi iterativi

3.7 Ottimizzazione nel caso di una variabile reale

1. Riconoscere l'esistenza di un massimo e di un minimo
2. Metodo di Newton per l'ottimizzazione
3. Approssimare il punto di minimo locale usando il metodo di Newton per l'ottimizzazione
4. Realizzare un un linguaggio di programmazione e al foglio di calcolo il metodo di Newton per l'ottimizzazione
5. Realizzare un un linguaggio di programmazione e al foglio di calcolo il metodo della sezione aurea

4 Conclusioni

Il percorso proposto mira al conseguimento degli obiettivi proposti dal Ministero dell'Istruzione e del Merito e, a giudizio dell'autore, forma degli studenti maggiormente consapevoli e capaci di orientarsi meglio nelle scelte universitarie.

Le tematiche proposte si sono dimostrate stimolanti sia per gli studenti che per l'insegnante.

L'esperienza è però difficile da riprodurre e richiede la produzione di tanto materiale didattico da fornire agli studenti.

Per poter osare con una didattica di questo tipo occorre arricchire di OER i progetti esistenti, come quello del Politecnico di Torino [12], di tradurre in modo professionale i migliori progetti didattici già attuati all'estero, di creare comunità di insegnanti di informatica.

References

[1] AA.VV. *Schema di regolamento recante 'Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento concernenti le attività e gli insegnamenti compresi nei piani degli studi previsti per i percorsi liceali di cui all'articolo 10, comma 3, del decreto del Presidente della Repubblica 15 marzo 2010, n. 89, in relazione all'articolo 2, commi 1 e 3, del medesimo regolamento.'*. Ministero dell'Istruzione, dell'Università e della Ricerca, 7 ottobre 2010. Allegato F.

[2] AA.VV. Asciidoctor. https://asciidoctor.org/, 2023.

[3] AA.VV. code.org. https://code.org/, 2023.

[4] AA.VV. Csforall. https://www.csforall.org/, 2023.

[5] AA.VV. Khan academy. https://www.khanacademy.org/, 2023.

[6] AA.VV. National centre for computer education. https://teachcomputing.org/, 2023.

[7] Harold Abelson and Gerald Jay Sussman. *Structure and interpretation of computer programs*. The MIT Press, 1996.

[8] Harold Abelson and Gerald Jay Sussman. *Structure and Interpretation of Computer Programs: JavaScript Edition*. MIT Press, 2022.

[9] Tim Bell, Ian H. Witten, Mike Fellows, and Renzo Davoli. *Computer Science Unplugged*. http://csunplugged.org/wp-content/uploads/2016/02/csunplugged-it.2015.1.0.pdf, 2016.

[10] College Board. Ap computer science principles. https://apstudents.collegeboard.org/courses/ap-computer-science-principles, 2023.

[11] Computer Science Education Research Group at the University of Canterbury, New Zealand. Computer science field guide. https://www.csfieldguide.org.nz/, 2023.

[12] Politecnico di Torino. Free architecture for remote education. https://fare.polito.it/, 2023.

[13] Kathi Fisler, Emmanuel Schanzer, Steve Weimar, Annie Fetter, K Ann Renninger, Shriram Krishnamurthi, Joe Gibbs Politz, Benjamin Lerner, Jennifer Poole, and Christine Koerner. Evolving a k-12 curriculum for integrating computer science into mathematics. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 59–65, 2021.

[14] Brian Fowler and Emiliana Vegas. How england implemented its computer science education program. *Center for Universal Education at The Brookings Institution*, 2021.

[15] Daniel P Friedman and Matthias Felleisen. *The Little Schemer*. MIT Press, 1995.