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

Gionata Massi

IIS Savoia Benincasa, Ancona (AN)

Sommario

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

Il curricolo rispetta i traguardi di competenza fissati dalle Indicazioni nazionali [11] ma presenta una selezione di argomenti e contenuti che sconcertano studenti, genitori e colleghi perché non rispondono alle attese di un percorso di addestramento all'uso delle tecnologie recenti.

Nel testo viene fornito un estratto della programmazione didattica nel primo e secondo biennio e nel quinto anno indicando un sottoinsieme di argomenti, la collocazione all'interno dell'area tematica definita nelle Indicazioni Nazionali, la motivazione principale per la scelta degli argomento, i traguardi di competenza espressi come abilità verificabili e i materiali didattici usati.

Il curricolo è orientato alla comprensione dei fondamenti epistemologici dell'informatica piuttosto che all'uso delle tecnologie ma richiede l'uso del PC, come implicitamente richiesto dalla Indicazioni Nazionali, e applicazioni rilasciate come software libero o, laddove non fosse possibile, a solo sorgente aperto.

La sperimentazione del curricolo è stata effettuata tra gli anni scolastici 2018/19 e 2022/23 coinvolgendo due classi e, soprattutto per la sopraggiunta emergenza sanitaria, non è stato possibile condurre analisi sul raggiungimento degli obiettivi di competenza prefissati confrontandosi con altre classi parallele.

In conclusione il curriculo qui proposto può essere utilizzato come un catalogo di traguardi di apprendimento dal quale gli insegnanti possono attingere liberamente per la progettazione e la programmazione della didattica dell'informaticadel proprio istituto e può essere oggetto di ulteriore sperimentazione.

Lo sviluppo futuro auspicato è quello della produzione da parte della comunità degli insegnanti della scuola secondaria di materiali di studio in lingua italiana e di elevata qualità, auspicabilmente nella forma di Open Educational Resource (OER) e con revisione da parte accademica, che rendano agevole l'adozione di un tale curriculo.

Indice

*neologismo trovato in alcuni lucidi del prof. Mattia Monga per indicare l'uso del computer o di applicativi specifici

[1] Harold Abelson nella prima lezione del corso MIT 6.001 Structure and Interpretation of Computer Programs sembra essere in disaccordo sulla natura scientifica dell'informatica

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 progettare e programmare l'attività didattica mediando fra vincoli normativi e disparate esigenze.

Una delle esigenze è quella di mediare con l'aspettativa dell'utenza, studenti e famiglie, e quello che richiedono con maggior frequenza è un percorso di applimatica*, ossia un percorso volto a fornire le istruzioni per l'utente di qualche programma applicativo. Anche i colleghi dei consigli di classe hanno le loro aspettative e, sempre nell'esperienza dell'autore, richiedono che l'insegnamento includa un certo programma per l'elaborazione del testo e un altro per la predisposizione dei lucidi di supporto alla presentazione. I docenti di matematica e fisica considerano anche altre categorie di software, in particolare i fogli di calcolo, e a volte sistemi per la rappresentazione geometrica e il calcolo numerico o simbolico. Sempre limitatamente all'esperienza dell'autore, si considerano solo applicativi con un'interfaccia di tipo *What You See Is What You Get*.

L'aspettativa sull'insegnamento dell'applimatica potrebbe anche essere condizionata dal fatto che molti istituti scolastici sono anche certificatori di alcune competenze nell'uso di software applicativi soprattutto per l'ufficio.

I libri di testo spesso si adeguano a queste aspettative, sicuramente per via delle Indicazioni Nazionali, e negli ultimi tre anni propongono un percorso di apprendimento di un linguaggio di programmazione[2], in genere uno e uno solo e frequentemente il C++[3].

Tra settembre e novembre, il gruppo di insegnanti di informatica di un liceo italiano, attenendosi alle Indicazioni nazionali e al PTOF della scuola in cui insegna, considerate le necessità del territorio, tenuto conto delle aspettative di studenti, genitori e colleghi, preso atto del libro di testo in adozione o della sua mancanza, progetta o riprogramma la sua programmazione didattica.

Il confronto con i curricola nelle nazioni anglofone, le uniche per le quali l'autore è riuscito ad effettuare una ricerca sullo stato dell'arte, inducono a ritenere che il nostro agire didattico

[2]Nel primo biennio si sta adottando sempre più frequentemente il termine *coding* e si sostituiscono i linguaggi di programmazione testuali con ambienti grafici per la programmazione visuale a blocchi

[3]Bjarne Stroustrup sembra sorpreso dal fatto che il C++ sia ampiamente utilizzato nella didattica in quanto tale linguaggio non è il più compatto e pulito e nella genesi del linguaggio non vi sono elementi sulla natura didattica.

per l'apprendimento dell'informatica sia disomogeneo, orientato all'uso delle applicazioni e non supportato da materiale di studio valido[4].

Negli Stati Uniti ci sono curricola e attività didattiche progettate in dettaglio, come [9] che propone scopi affini alle nostre *indicazioni nazionali* [11].

Le esperienze didattiche e i curricula sono spesso raccolti in cataloghi come [14] e vengono offerti vari corsi, anche gratuiti, come quelli sulle piattaforme Code.org [13] e Khan Academy [12], basati sui curricula più noti, come lo Advanced Placement Computer Science [5].

La comunità scientifica appare da tempo attiva nel formulare proposte didattiche che includono lo sviluppo di linguaggi di programmazione e ambienti per l'insegnamento[5] ([1] e [2], [8]).

In Nuova Zelanda hanno sviluppato la *Computer Science Field Guide* [7], 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[6] [10].

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

[4] Il sistema scolastico italiano ha molte peculiarità che lo rendono estremante formativo, o almeno questa è l'impressione che se ne ricava quando si assiste ai colloqui pluridisciplinari per gli studenti che hanno frequentato un anno scolastico all'estero, ma questo non si riesce ad asserire per l'informatica.

[5] Seymour Papert e Mitchel Resnick sono così famosi tra gli insegnati italiani che è inutile citarli ma il loro contributo ora viene usato per spingere verso il *coding* che pare cosa diversa dalla programmazione.

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

[7] 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

Si è scelto AsciiDoctor[3] con LaTeX e `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' [4].

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 Sperimentazione

L'azione didattica ispirata da tale curricolo è stata condotta tra gli anni scolastici 2018/19 e 2022/23 coinvolgendo due classi, di cui una di una sperimentazione quadriennale. Entrambe le classi hanno avuto come insegnante, dal primo all'Esame di Stato, l'autore di questo contributo.

Le misure adottate per fronteggiare l'emergenza sanitaria per COVID-19 nel periodo in cui si è adottato il curricolo hanno condizionato negativamente sulla sperimentazione e sulla rilevazione dei risultati. È stato impossibile condurre un'analisi sul raggiungimento degli obiettivi di competenza per classi parallele al termine del secondo anno, periodo in cui gli studenti non erano nelle aule scolastiche, e non si è proceduto al rilevamento nel quarto e quinto anno per via della forte eterogeneità con le classi parallele che si è avuta anche per l'assenza di didattica.

L'esperienza si è conclusa per l'esigenza di uniformare l'offerta didattica tra classi parallele, per la difficoltà di reperire validi supporti didattici e per la scarsità del tempo che l'autore ha trovato per la produzione del materiale didattico.

5 Conclusioni

In conclusione si considera il curriculo qui proposto come una bozza per la progettazione e la programmazione della didattica dell'informatica alla quale gli insegnati liceali possono attingere per la costruzione del percorso del proprio istituto.

Non si possono fornire indicazioni sull'efficacia della proposta a causa della mancanza di uno studio comparativo.

Si rileva la mancanza di materiali di studio, auspicabilmente nella forma di Open Educational Resource (OER), in lingua italiana e di qualità elevata, che sono supporti indispensabili per gli studenti.

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 [15], di tradurre in modo professionale i migliori progetti didattici già attuati all'estero, di creare comunità di insegnanti di informatica.

Riferimenti bibliografici

[1] Harold Abelson e Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. Versione *open access*: https://sarabander.github.io/sicp/. The MIT Press, 1996, p. 683. ISBN: 9780262510875.

[2] Harold Abelson et al. *Structure and Interpretation of Computer Programs: JavaScript Edition. JavaScript Edition*. Versione *open access*: https://sourceacademy.org/sicpjs/index. MIT Press, 2022. ISBN: 9780262543231.

[3] Dan Allen. *Asciidoctor*. 2023. URL: https://asciidoctor.org/.

[4] Tim Bell et al. *Computer Science Unplugged*. http://csunplugged.org/wp-content/uploads/2016/02/csunplugged-it.2015.1.0.pdf, 2016.

[5] College Board. *AP Computer Science Principles*. 2023. URL: https://apstudents.collegeboard.org/courses/ap-computer-science-principles.

[6] National Centre for Computer Education. *Helping you teach computing*. 2023. URL: https://teachcomputing.org/.

[7] Computer Science Education Research Group at the University of Canterbury, New Zealand. *Computer Science Field Guide*. 2023. URL: https://www.csfieldguide.org.nz/.

[8] Matthias Felleisen et al. *How to Design Programs. An Introduction to Programming and Computing*. Versione *Open Access*: https://htdp.org/. MIT Press, 2018, p. 792. ISBN: 9780262534802.

[9] Kathi Fisler et al. «Evolving a K-12 Curriculum for Integrating Computer Science into Mathematics». In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. SIGCSE '21. Virtual Event, USA: Association for Computing Machinery, mar. 2021, pp. 59–65. ISBN: 9781450380621. DOI: 10.1145/3408877.3432546. URL: https://doi.org/10.1145/3408877.3432546.

[10] Brian Fowler e Emiliana Vegas. «How England Implemented Its Computer Science Education Program.» In: *Center for Universal Education at The Brookings Institution* (2021).

[11] Il Ministro dell'Istruzione, dell'Università e della Ricerca. *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.'* Allegato F. Ministero dell'Istruzione, dell'Università e della Ricerca. Lug. 2010. URL: https://www.gazzettaufficiale.it/eli/id/2010/12/14/010G0232/sg.

[12] Sal Khan. *Khan Academy*. 2023. URL: https://www.khanacademy.org/.

[13] Code.org Team. *Code.org*. 2023. URL: https://code.org/.

[14] CSforALL Team. *Computer Science for **ALL** Students*. 2023. URL: https://www.csforall.org/.

[15] Politecnico di Torino. *Free Architecture for Remote Education*. 2023. URL: https://fare.polito.it/.