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 [6] 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*.

Due classi, di cui una di un percorso quadriennale, hanno seguito questo curricolo per l'intero ciclo e permettono all'autore di affermare che è possibile superare e due proposte didattiche dominanti la didattica dell'informatica nel liceo scientifico a fronte di un enorme lavoro del docente scarsamente supportato da materiale didattico pensato per la scuola italiana.

La sperimentazione, infatti, 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 onerosa senza un adeguato deposito di Open Educational Resource (OER), in lingua italiana e di qualità, possibilmente revisionato dalla comunità scientifica, che possa agevolare gli insegnati, fornire un supporto allo studio e che permettere l'unificazione della terminologia usata in ambito informatico.

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 godono di sillabi o programmazioni nazionali e non hanno tecnologie sviluppate per l'insegnamento della loro materia ma devono costruirsi una programmazione didattica mediando fra varie vincoli normativi e svariate esigenze.

Negli Stati Uniti, ad esempio, ci sono tanti progetti con curricoli e di attività didattiche progettate fin nei minimi particolari, tra cui il curricolo presentato in [4] che è analogo negli scopi a quanto stabilito nelle indicazioni nazionali [6]. Le strategie didattiche sono spesso influenzate dai concetti inerenti la programmazione e per quest'ultima vari comunità scientifiche propongono strategie e linguaggi didattici soprattutto nello MIT[3] ([2] e [3]) o stili narrativi particolarmente innovativi ([5]). Alcuni siti raccolgono le esperienze e i curriculum, come [1].

In Nuova Zelanda

In Gran Bretagna

Ci si deve attenere alle Indicazioni nazionali e al PTOF, si devono considerare le necessità del territorio, ci si dovrebbe adeguare alle aspettative di studenti, genitori e colleghi e, se si adotta un testo, ci si deve attenere alle scelte degli autori.

Il curricolo proposto in questo contributo è stato progettato rilassando gli ultimi due vincoli: sconvolge le aspettative e non prevede libri. Le attività sono svolte prevalentemente al PC e alla lavagna usando esclusivamente software libero e formati aperti.

2 Primo biennio

Senza soffermarsi sugli tutti gli obiettivi specifici di apprendimento previsti per il primo biennio, si propongono alcune attività apparentemente inappropriate contestualizzandole nelle indicazioni nazionali.

2.1 Scrittura tecnico-scientifica al calcolatore

Per soddisfare il requisito: '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[4], 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

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

[4] 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

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

Come strumento si è scelto AsciiDoctor integrato di LaTeXper la matematica e vari tool inclusi in `asciidoctor-diagram`. Come piattaforma di verifica l'autore ha costruito un'applicazione web basata su `Asciidoctorjs` per erogare il compito, inteso come elenco di specifiche, e verificare il loro soddisfacimento tramite l'analisi dell'oggetto analizzato dalla libreria. Le formule matematiche e i diagrammi sono, invece, stati valutati tramite espressioni regolari. La didattica del linguaggio è stata condotta con esempi e generalizzazioni della sintassi tramite diagrammi rail-road.

2.2 Dati e codifiche

Per soddisfare il requisito: '... una introduzione alla codifica binaria ... i codici ASCII e Unicode', si è proposta un'unità didattica di introduzione alle codifiche per lo sviluppo delle seguenti abilità:

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 per eccellenza è stato il testo 'Computer Science Unplugged' con le sue carte e le schede per le attività. La verifica è stata erogata via Moodle usando un quiz con domande causali scelte su depositi di domande a risposta numerica per le conversioni di base, a risposta chiusa sulle codifiche di parole[5], immagini di caratteri da un font creato dall'autore.

[5] Si sono estratte da una pagina di Wikipedia tutte le parole di lunghezza compresa tra 5 e 7 caratteri aventi al massimo due caratteri uguali e che non presentassero le lettere J, W, X, Y, Z

2.3 Problemi, modelli, soluzioni e algoritmi

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 applicato ad una sequenza numerica
6. Simulare l'algoritmo di ordinamento per inserimento applicato ad una sequenza numerica
7. Simulare l'algoritmo di ordinamento a bolle applicato ad una sequenza numerica
8. Calcolare il numero di confronti e di scambi degli algoritmi di ordinamento basati su confronti e scambi
9. Comprendere i criteri di scelta di un algoritmo rispetto ad altri
10. Astrarre il modello di semplici problemi di natura quantitativa e descrivere algoritmicamente il procedimento di soluzione
11. Simulare l'esecuzione di un programma

Si sono usati gli esempi su 'Computer Science Unplugged'.

2.4 Interagire col sistema operativo tramite la shell

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

3 Secondo biennio e quinto anno

3.1 Documenti elettronici per il web

1. Produrre una pagina web in HTML usando i marcatori più comuni
2. Rappresentare gli elementi dell'albero DOM di un documento HTML

3.2 Fogli elettronici

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

3.3 Paradigmi di programmazione

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 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 gli algoritmi di Prim e Kruskal
6. Simulare l'algoritmo di Dijkstra
7. 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. Le tematiche scelte sono stimolanti per l'insegnante e per gli studenti. A volte si trovano smarriti rispetto alle cose che gli sono più famigliari ma poi le trovano più utili e interessanti di quello che già conoscono. Gli aspetti più formali e astratti sono difficili da cogliere senza uno studio attento e rigoroso ma nella programmazione non vi sono concetti più astratti o complessi di quelli che si affrontano nella matematica, nella fisica o nelle scienze. Si può osare una programmazione di questo tipo ma si rischia di fare una fatica immane per la mancanza di materiale scritto per studenti della scuola superiore. Servirebbero libri di testo e progetti come csfieldguid, o i vari libri di matematica in licenza creative commons.

References

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

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

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

[4] 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.

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

[6] Ministero 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.'*, 7 ottobre 2010. Allegato F.