Consegna 6
la natura dei programmi

Lorenzo Dentis, lorenzo.dentis@edu.unito.it

31 marzo 2023

1 Domande

Partendo dal materiale relativo al seminario tenuto dalla Dott.ssa Violetta Lonati Di cosa parliamo quando parliamo di programmi scrivete una breve riflessione su questi aspetti:

- Nei corsi di studio che avete affrontato sono emerse tutte le sfaccettature del concetto di programma presentate nel seminario?
- Guardando le indicazioni nazionali del laboratorio Informatica e Scuola del cini e quelle degli istituti superiori vi sembra che includano tutte le 6 facce dei programmi?
- Dovendo progettare un intero corso di scienze informatiche in una scuola superiore in che ordine presentereste le 6 sfaccettature?

2 Risposte

2.1 1 Esperienze personali

Ho frequentato il liceo scientifico delle scienze applicate e ho avuto la fortuna di avere un buon insegnante di informatica. Abbiamo coperto quasi tutti i sei campi, ma ci siamo concentrati principalmente sui programmi come oggetti nozionali, approfondendo molto il linguaggio C ad esempio, e sui programmi come oggetti astratti, con molta enfasi sulla differenza tra algoritmo e codice e sul problem solving piuttosto che sulla scrittura di codice. È stato dato molto spazio anche al tema dei programmi come entità eseguibili, poiché abbiamo fatto molta programmazione.

Contrariamente a quanto si potrebbe pensare, non abbiamo visto molto il programma come strumento, ricordo pochi riferimenti alle competenze digitali o all'informatica in ambito interdisciplinare. Tuttavia, abbiamo utilizzato strumenti a supporto dello sviluppo di algoritmi o della scrittura di codice, come Scratch, Algobuild e vari IDE.
Non è stata quasi affrontata la visione del programma come opera dell'uomo, ad eccezione di una lezione in cui sono stati affrontati aspetti del ruolo dell'informatica nella storia, parlando di Alan Turing e della macchina Colossus. Non è stato affrontato il tema del programma come oggetto fisico, nonostante una

breve introduzione all'hardware, all'architettura e alle reti, ma mai in relazione ai programmi. Il professore ha però proposto una attività facoltativa durante una autogestione riguardo alla programmazione di microcontrollori (Arduino)

In ambito universitario sono state ampiamente approfondite tutte e sei le sfaccettature.

2.2 3 Ordine di presentazione

Dovendo scegliere proporrei questo ordine:

1. Entità astratte
2. Artefatti linguistico-notazionali
3. Entità eseguibili
4. Opere dell'uomo
5. Strumenti
6. Oggetti fisici

Il primo passo è capire cosa sia un algoritmo e come possa essere usato per risolvere problemi. È possibile utilizzare esempi di problemi, non necessariamente problemi reali, per illustrare il concetto.**Programma come entità astratta**. Successivamente, è possibile passare dal modello "pseudocodice" alla codifica, utilizzando anche un linguaggio semplice come Scratch. Questo serve sia per introdurre un po' di azione pratica, che secondo il costruttivismo aiuta a ricordare meglio le nozioni, sia per mantenere vivo l'interesse degli studenti, fattore da non sottovalutare. **programma come artefatto linguistico-notazionale**.

Una volta introdotto l'algoritmo e un linguaggio di programmazione, si può pensare di eseguire il programma, anche presentando un debugger o una metodologia di logging, come delle print in mezzo al codice per comprendere lo stato delle variabili durante l'esecuzione. **programma come entità eseguibile**

Dopo aver compreso cosa sia un algoritmo e aver scritto programmi semplici, è importante presentarli come **opere dell'uomo** e spiegare che hanno senso solo nel contesto in cui risolvono un problema specifico e che soprattutto sono limitati secondo i requisiti. Si possono presentare problemi reali e strutturati e proporre programmi per la loro risoluzione, magari anche scritti dagli studenti, che possano essere utilizzati come strumenti. Un esempio classico è la calcolatrice che permette di risolvere equazioni di secondo grado con la formula del delta. Oppure si possono utilizzare programmi in ambito interdisciplinare, mostrando il programma come **strumento**. Infine, si può presentare la visione del programma come **oggetto fisico**, spiegando l'architettura hardware e alcuni concetti di come funzionano CPU e memoria, sempre con l'ausilio di esercizi.

Sviluppando questa risposta, mi sono reso conto di quanto sia utile in realtà portare avanti tutti gli aspetti in parallelo, almeno i primi 3-4. Mentre si presenta la struttura di un algoritmo, si può iniziare a mostrare i primi costrutti in codice, anche con esercizi di lettura del codice. Una volta acquisite le basi, si può già pensare di eseguire programmi, senza necessariamente completare i

primi punti. Allo stesso modo, si possono fare accenni alla memoria, alla complessità o allo scopo di un programma, anche se gli studenti non sono ancora in grado di scriverne uno completo.

In sintesi, l'approccio suggerito è quello di uno sviluppo incrementale, piuttosto che "waterfall", prestando comunque attenzione alle priorità che ho esposto.