Educare al pensiero computazionale: alcuni approfondimenti e relativi apporti formativi

Seconda parte

MICHELE PELLEREY[1]

Questo secondo contributo sulla tematica del pensiero computazionale in continuità con il precedente intende approfondirne alcune componenti concettuali e prospettare una quadro di obiettivi formativi che dovrebbero costituire mete fondamentali da conseguire nei percorsi di Istruzione e Formazione Professionale. Quanto viene prospettato dalla recente Raccomandazione europea sulla competenze chiave per l'apprendimento permanente relativa alle competenze digitali e all'alfabetizzazione informatica esige lo sviluppo di un'identità professionale adeguata alle trasformazioni tecnologiche e organizzative che stanno contraddistinguendo i nostri tempi.

This second article on the topic of computational thinking is the continuation of the article published in the previous issue. It aims at deepening some conceptual aspects and at presenting a framework of training objectives that should constitute the main goals to be achieved in the paths of Vocational Education and Training. The recent European Recommendation on key competences for lifelong learning related to digital skills and computer literacy states that it is necessary to develop a professional identity in line with the technological and organizational transformations that are distinguishing our times.

1. Programmazione e livelli di astrazione

Nel precedente contributo[2] è stato evidenziato come nel pensiero computazionale gioca un ruolo essenziale il processo di rappresentazione astratta di una realtà concreta, sia essa una situazione o un procedimento. Ora occorre precisare che tale processo si svolge spesso a livelli crescenti di astrazione al fine di dominare razionalmente l'essenza o il nucleo portante del problema e della possibile soluzione da individuare. Per capire questa esigenza si può esaminare un po' più da vicino ciò che avviene nell'attività di programmazione di un software applicativo.

[1] Professore emerito, già Ordinario di Didattica dell'Università Pontificia Salesiana di Roma.

[2] PELLEREY M., Educare al pensiero computazionale: un'esigenza per i processi di Formazione Professionale oggi. Prima parte, *Rassegna CNOS*, 2018, 2, pp. 37-51.

Possiamo partire da un'osservazione generale: quanto avviene a livello di esecuzione da parte dell'elaboratore elettronico è ormai da noi inafferrabile. A livello dell'esecutore automatico, le operazioni da compiere, come i dati da conservare, sono in realtà stringhe di bit (binary digit), cioè insiemi ordinati di 0 e 1. Questi bit normalmente sono raggruppati in ottetti formando i cosiddetti byte. Di conseguenza le cifre numeriche, i caratteri alfabetici e i vari segni grafici che compongono informazioni e comandi devono essere rappresentati mediante sequenze di bit. A questo fine viene normalmente utilizzato il cosiddetto codice ASCII (American Standard Code for Information Interchange, codice standard americano per lo scambio di informazioni). La ragione dell'uso del bit come base di codificazione sta nel fatto che esso può corrispondere sul piano fisico o a uno stato magnetico di polarizzazione, o a uno stato di tensione elettrica.

Così anche la potenza di un computer può essere misurata in byte: kilobyte (1.024 byte, KB), megabyte (1.048576 byte, MB), terabyte (1.099.511.627.776 byte, TB), ecc., sia come potenza di calcolo, sia come ampiezza di memoria. Un linguaggio di comunicazione comprensibile direttamente da parte di una macchina esecutrice dovrebbe fare i conti con scritture assai lontane dalle forme linguistiche consuete. Fortunatamente nel tempo sono stati sviluppati opportuni *compilatori*, specie di traduttori di parole e di cifre scritte in linguaggi più consueti per noi, in forme linguistiche i cui caratteri sono formati da stringhe di bit. Sono stati anche sviluppati linguaggi che si presentano assai lontani da quelli comprensibili da una macchina i cosiddetti *linguaggi evoluti* o *ad alto livello*, più vicini al modo di pensare e scrivere un procedimento risolutivo da parte di un essere umano. Questi linguaggi utilizzano codici, detti sorgente, che devono però essere tradotti in codici, detti oggetto, direttamente comprensibili ed eseguibili dalle macchine. Cioè, per poter essere comunicati efficacemente a un esecutore elettronico questi linguaggi devono essere tradotti automaticamente in un *linguaggio* cosiddetto *di basso livello*, ossia comprensibile dalla macchina.

Dal punto di vista dell'esecutore fisico, quindi, si hanno così linguaggi sempre più astratti e lontani dalla sua possibilità di comprensione. Il poterli utilizzare efficacemente deriva dall'aver saputo costruire una cascata o catena di traduttori che interpretano e riportano progressivamente da un livello astratto, lontanissimo spesso da quanto compreso dalla macchina, a un livello di immediatezza e comprensibilità da parte di questa e così poter essere eseguiti. Dal punto di vista della macchina questo è un processo di semplificazione del linguaggio astratto usato, mentre da quello del programmatore la semplificazione avviene in senso inverso. Ma con una conseguenza assai rilevante: un'aumentata facilità di errori, in termini tecnici detti *bug*. Questo, perché il linguaggio evoluto non cessa di dover essere rigidamente strutturato nella sua grammatica e nella sua sintassi. Ad esempio, un errore di sintassi può non essere avvertito da chi ela-

bora il programma, ma l'esecutore ne è certamente influenzato: così il programma può non funzionare. Esistono anche errori di semantica, nel senso che il computer riesce a capire quanto richiesto, ma in senso diverso da quello inteso dal programmatore e il risultato allora non è quello atteso.

Correggere con cura e precisione un programma è un'attività fondamentale, detta di *debugging,* che si attua in dialogo continuo con l'esecutore. Questo fornisce infatti un sistematico feedback, evidenziato dal suo non funzionamento o funzionamento errato. Si tratta di un feedback che viene definito "feedback interno o intrinseco", perché non viene dato da altri che dalla risposta che viene dal tentare di far eseguire dalla macchina il procedimento programmato. Tutto ciò da un punto di vista educativo è di grande importanza non solo sul piano dell'apprendere a programmare, ma anche più in generale. Tanto è vero che ne è nata una corrente di psicologia dell'educazione, detta del "costruzionismo", che studia la costruzione della conoscenza in dialogo con i risultati delle proprie azioni. Una forma di autovalutazione continua del proprio agire e dei suoi risultati che assume una valenza formativa assai elevata[3].

Inoltre, il dover scrivere seguendo rigidamente regole grammaticali e sintattiche precise può aiutare a migliorare la propria maniera di scrivere in generale, ma soprattutto nella direzione della redazione di testi logici e coerenti.

2. Un approfondimento sul pensiero algoritmico

Una branca di studio assai sviluppata e feconda propria del pensiero computazionale è quella dedicata all'elaborazione di una teoria degli algoritmi, uno studio sistematico sulla loro natura e articolazione. A esempio, uno dei teoremi fondamentali individuati e dimostrati a Roma da Corrado Boehm e Giuseppe Jacopini nel 1966 è che qualsiasi algoritmo, anche il più complesso, si basa in definitiva su tre schemi o strutture elementari di controllo: la sequenza, la selezione, l'iterazione o ciclo. È importante, dunque, esaminare con cura questi schemi algoritmici per saperli utilizzare quando necessario od opportuno. Per far questo utilizzeremo quello che è stato denominato *linguaggio di progetto* o *pseudocodice*, una forma di rappresentazione che mette bene in luce la struttura dell'algoritmo facilitando da una parte la sua comprensione, dall'altra, la sua traduzione successiva in un linguaggio evoluto. Un'altra forma ormai assai comune di rappresentazione di un algoritmo è quella che utilizza i diagrammi di flusso.

[3] Si è approfondito l'apporto educativo del costruzionismo nel seguente contributo: M. PELLEREY, Verso una più diffusa e incisiva valorizzazione di un apprendimento basato sulla pratica, anche in un ambiente digitale, *Rassegna CNOS*, 2015(31), 3, pp. 57-68.

a) La *sequenza* è una successione ordinata di istruzioni che devono essere eseguite in maniera puntuale per ottenere il risultato previsto. Si tratta del più semplice e immediato schema algoritmico. Ad esempio per calcolare il perimetro e l'area di un quadrato i passi successivi da compiere sono:

Per perimetro	Per area
Scrivi la misura del lato	Scrivi la misura del lato
Moltiplica il lato per 4	Moltiplica il lato per se stesso
Scrivi il risultato	Scrivi il risultato

b) La *selezione* è costituita dalla scelta che deve essere fatta tra due percorsi possibili in base a una condizione che può risultare vera o falsa. Questa struttura può essere rappresentata opportunamente, a esempio descrivendo l'attraversamento del passaggio a livello di una ferrovia controllato da semaforo e sbarre:

Passaggio a livello **se** rosso e sbarre abbassate **allora** aspetta **altrimenti** attraversa con cautela.

c) L'*iterazione* o *ciclo*: è costituita da una successione di operazioni da compiere finché non cambia una precisa condizione. Ecco l'esempio di ricerca e stampa dell'indirizzo di un nome su un elenco:

Per ricerca e stampa di un indirizzo leggi un primo nome dell'elenco, **se** è il nome cercato **allora** stampa l'indirizzo **altrimenti** leggi il nome successivo **finché** trovi in nome cercato o è esaurito l'elenco.

In generale schemi algoritmici anche abbastanza semplici includono più di uno schema elementare. D'altra parte molti schemi algoritmici, anche perché spesso utilizzati, diventano componenti di algoritmi più complessi e per questo vengono chiamati *moduli* e identificati con una propria denominazione. Ad esempio ecco lo schema del contatore, cioè lo schema iterativo che ripete l'operazione "più uno" successivamente a partire da zero, finché…si vuole o è necessario.

Modulo contatore, parti da 0, **ripeti** + **1 finché** necessario.

Come precedentemente accennato una delle attività fondamentali di un programmatore è quella di progettare e realizzare un programma che funzioni correttamente. Per questo è continuamente impegnato nel controllarne sia la correttezza, sia la complessità, al fine di migliorarlo o correggendo gli errori o semplificandolo finché ciò è possibile. Molti degli aggiornamenti delle app che compaiono sul nostro smartphone derivano proprio da questi lavori di ottimizzazione del software.

L'utilizzo della rappresentazione degli algoritmi, ad esempio mediante opportuni diagrammi di flusso, è assai utile nell'analisi e comprensioni di molti procedimenti presenti nella matematica come in altri ambiti di conoscenze disciplinari, soprattutto quando si tratta di classificazioni e ordinamenti. Dal punto di vista professionale forse il contributo maggiore viene dal dover evidenziare bene la sequenza delle operazioni da compiere nei vari procedimenti produttivi,

in particolare quando si ha a che fare con macchine e strumenti digitali o informatici. Il controllo attento dei vari passi operativi contribuisce non poco alla stessa sicurezza nel lavorare con macchinari spesso complessi, il cui attento controllo risulta essenziale per un buon andamento dell'attività.

3. Un approfondimento sulle basi di dati

La progettazione, realizzazione e valorizzazione di una base di dati coinvolge alcuni apporti fondamentali che provengono dalla matematica e dalla logica: il concetto di variabile e quelli di classificazione e ordinamento. Il concetto di variabile sembra a prima vista abbastanza intuitivo: si tratterebbe di considerare una caratteristica degli oggetti o dei fenomeni che varia nel tempo come la temperatura durante la giornata, l'altezza o il peso di un bambino che cresce. Pur potendosi appoggiare a questo tipo di intuizione, in realtà in matematica la variabile normalmente indicata da una lettera dell'alfabeto, a esempio *a*, è in realtà un segna posto. Essa indica, cioè, uno posto nel quale può essere inserito un numero, o una misura, che varia in un insieme dato: al posto della lettera *a* può essere inserito un numero naturale, cioè un elemento che varia nell'insieme dei numeri naturali, oppure un numero decimale, cioè un elemento variabile dell'insieme dei numeri decimali. Il vantaggio di utilizzare lettere al posto di numeri precisi, sta nel fatto che si possono così indicare strutture matematiche generali, applicabili a molti casi specifici. A esempio una formula come quella di calcolo dell'area di un quadrato può essere utilizzata per quadrati di grandezza e misura qualsiasi: $A = l^2$. Questo vale in generale per tutte le formule che esprimo leggi fisiche, leggi chimiche o di scienze naturali.

Nell'ambito dell'informatica una variabile può essere pensata come una casella vuota, alla quale dobbiamo dare un nome e nella quale possiamo collocare dati di vario tipo. Se dobbiamo comunicare al computer di calcolare l'area di un rettangolo qualsiasi, possiamo denominare "base" la casella nella quale saranno collocati i valori da assegnare alla base del rettangolo, "altezza" la casella nella quale saranno collocati i valori da assegnare all'altezza del rettangolo. In altre parole definiamo due variabili denominate "base" e "altezza" alle quali verranno assegnati particolari valori. Nel linguaggio di progetto precedentemente utilizzato l'assegnazione a una casella, denominata a esempio K, di un valore della variabile considerata (numerica, alfabetica o alfanumerica), si può indicare nel modo seguente: K:= 1, oppure K 1 (assegnazione del valore numerico 1 alla variabile K).

L'insieme dei dati, che si intendono raccogliere, conservare e utilizzare nella memoria di un elaboratore, deve essere opportunamente organizzato, o struttu-

rato, nella prospettiva di una loro valorizzazione. Così una branca del pensiero computazionale si occupa della possibili strutture di dati e della loro più o meno bontà ai fini delle future utilizzazioni. Entrare nei dettagli delle possibili strutturazioni dei dati esula da questa riflessione. Tuttavia è utile menzionarne almeno alcuni elementi per capire l'importanza di questo ambito della riflessione umana.

Nel contributo precedente è stata esaminata la struttura del codice fiscale di una persona fisica. Esso rappresenta quello che viene chiamato un *record*, cioè una sequenza o stringa di caratteri sia numerici, sia alfabetici, sia alfanumerici. Ciascuna casella di questa sequenza viene denominata *campo*. Nel codice fiscale sono presenti i seguenti campi, ciascuno contenente una variabile opportunamente codificata: cognome, nome, data di nascita e sesso, luogo di nascita, codice di controllo. Analogamente l'IBAN, il codice bancario, ha la struttura di un record con vari campi. Se un record è composto di variabili tutte dello stesso tipo (a esempio la lista dei voti presi a fine anno scolastico nelle varie materie) esso viene chiamato *vettore o array*.

L'insieme dei record che entrano a far parte della base di dati, ad esempio l'insieme dei codici fiscali degli italiani, deve essere, come sopra accennato, a sua volta strutturato, sempre sulla base delle possibili applicazioni. I record possono formare una lista ordinata sequenzialmente secondo un criterio predefinito (a esempio nel caso dei codici fiscali per ordine alfabetico), ma possono essere utili anche altri modelli organizzativi intuitivamente comprensibili sulla base della loro denominazione: modello gerarchico, modello reticolare, modello relazionale. È abbastanza intuitivo comprendere quanto sia importante progettare in maniera accurata e funzionale sia la struttura del singolo record, sia la struttura generale della base di dati, ma sempre a partire da ciò che ci si propone di ottenere. Si tratta di un processo di rappresentazione astratta, analogo per molti versi a quello che fa passare dalla considerazione di un'operazione direttamente riferita a numeri precisi a una, invece, riferita a variabili.

Naturalmente tale massa di dati per essere poi valorizzata ha bisogno di un sistema che permetta non solo di richiamare i record o i gruppi di record utili o necessari, di collegarli con altri apporti, di aggiornarli, di elaborarli opportunamente, ecc.: si tratta di quello che è stato denominato Sistema di Gestione della Base di Dati (Data Base Manager System). E qui entra in gioco di nuovo il pensiero algoritmico.

Ogni attività produttiva o commerciale ormai non può più fare meno dell'apporto della logistica, cioè di quella che è stata definita l'arte e la scienza il cui ambito di analisi riguarda: «l'insieme delle attività organizzative, gestionali e strategiche che governano nelle aziende i flussi di materiali e delle relative informazioni dalle origini presso i fornitori fino alla consegna dei prodotti finiti

ai clienti e al servizio post vendita»[4]. E la logistica oggi è essenzialmente legata alla progettazione, gestione e valorizzazione di opportune basi di dati. Di qui la grande importanza di una buona iniziazione da parte di ogni qualificato a questo ambito del pensiero computazionale.

Tuttavia, oggi più in generale occorre ricordare che l'impegno nell'organizzazione della massa di informazioni che viene raccolta sia nelle indagini di tipo scientifico, sia di tipo commerciale, sia di tipo produttivo, sia di tipo economico-finanziario è tale che senza una complessa attività di progettazione e strutturazione di adeguate basi di dati tutto ciò sarebbe poco proficuo. Inoltre l'uso di algoritmi anche assai sofisticati sembra poter dare spazio a catene decisionali, che condizionano non poco sia i mercati finanziari, sia quelli commerciali. In qualche maniera ogni nostra azione e interazione mediata dal computer (sia esso smartphone o altro) può essere registrata, trasformata in dato, collegata a un record complesso riferito alla nostra identità e utilizzata per sollecitare, se non pilotare, le nostre azioni future.

Da un punto di vista formativo si può anche osservare come questo ambito di studio permetta di sviluppare la capacità di raccogliere, conservare e valorizzare in maniera ordinata e funzionale ogni tipo di documentazione, informazione, testo utile o necessario, costituendo e/o sfruttando in maniera sistematica archivi sia cartacei, sia fisici, sia digitali, di vario genere: dai registri scolastici e formativi, alle biblioteche, alle emeroteche, ai musei.

Si può anche accennare a quella che viene denominata struttura cognitiva della persona. Anche nel caso del pensiero umano le informazioni che possiamo raccogliere mediante i nostri sensi, o direttamente o indirettamente mediante la lettura o altre forme di comunicazione mediata, per poterle conservare nella nostra memoria in maniera significativa e fruibile nel futuro, occorre che siano organizzate in maniera opportuna. Ciò viene fatto a livello di quella che viene denominata in psicologia memoria di lavoro. Già a livello di scuola dell'infanzia le esperienze dirette del bambino possono essere organizzate intorno alle parole, che diventano il perno di riferimento per poterle in seguito evocare. Analogamente le varie rappresentazioni simboliche, come quelle matematiche, diventano elementi intorno a cui si possono organizzare ulteriori esperienze. I processi di elaborazione, cioè di comprensione, e quelli di organizzazione della varie conoscenze di natura concettuale o procedurale sono essenziali per sviluppare una buona struttura cognitiva.

[4] È questa la definizione dell'Associazione Italiana di Logistica.

4. Sul concetto di automa e di reti di automi

Si è accennato nel contributo precedente al sogno millenario, concretizzato mito ebraico del Golem, cioè di una realtà costruita dall'uomo, dotata di una straordinaria forza e resistenza in grado di eseguire alla lettera gli ordini del suo creatore, di cui diventava una specie di schiavo, tuttavia incapace di pensare, di parlare e di provare qualsiasi tipo di emozione perché privo di un'anima. Si tratta di quello che oggi passa sotto la dizione "automa" esecutivo, un congegno che riesce a eseguire operazioni o movimenti propri dell'uomo ma privo di autonomia decisionale. Nel pensiero computazionale la nozione di automa cioè di esecutore in grado di portare a termine in modo effettivo un procedimento convenientemente a lui comunicato costituisce uno dei suoi pilastri concettuali. Tale esecutore può essere un uomo o una macchina, in quanto dal punto di vista concettuale è essenziale la considerazione "dell'agente elaboratore di informazioni (o per usare un termine più semplice, dell'esecutore). Senza l'esecutore e la sua capacità di operare in modo effettivo, non c'è informatica".[5] Per questo viene anche detto che il paradigma di riferimento dell'informatica non è tanto "risolvere problemi", quanto "far risolvere i problemi" a un esecutore. Inoltre "la formalizzazione dell'esecutore e delle risorse (tempo, spazio) a sua disposizione per l'effettuazione della computazione ha permesso di caratterizzare in modo esatto la «complessità» della risoluzione dei problemi e di determinarne la gerarchia, dai più semplici ai più difficili"[6]. Sulla base di queste conquiste scientifiche, le tecnologie crittografiche permettono di garantire la sicurezza delle transazioni della nostra carta di credito quando eseguiamo acquisti on line. E ancora, la formalizzazione di come l'esecutore rappresenta i dati a sua disposizione, le relazioni tra essi ed i relativi processi di elaborazione, ha consentito la definizione precisa dei processi con cui l'esecutore può "apprendere" e quindi, a esempio, fornirci quello specifico annuncio promozionale calibrato sui nostri interessi.

Naturalmente il termine "automa" rimanda a forme automatiche di comportamento, anche umano. Tuttavia l'evoluzione di questi ultimi decenni degli studi in merito ha portato in primo luogo alla distinzione tra automi in grado di eseguire un solo tipo di procedimento, identificando quindi l'automa con il procedimento stesso, ad automi cosiddetti universali, in grado cioè di eseguire qualsiasi procedimento. I computer attuali sono appunto detti strumenti universali perché il loro hardware, cioè la loro struttura, consente di eseguire molteplici

[5] Lodi M. – Martini S. – Nardelli E., *Abbiamo davvero bisogno del pensiero computazionale?* Mondo digitale, 2017, Novembre, p. 6.

[6] Ibidem, p. 7.

procedimenti specificati da opportuni software. In genere la macchine a controllo numerico si presentano come automi in grado di eseguire particolari tipologie di procedimenti lavorativi. In questi casi si tratta di automi reali, cioè di macchine dotate di una specifica struttura fisica. Tuttavia dal punto di vista del pensiero computazionale si possono, ed è essenziale spesso farlo, considerare automi ideali. Questi tendono a costituire riferimenti fondamentali per lo sviluppo teorico e in molti casi anche concreto dell'informatica. Ad esempio la concettualizzazione della cosiddetta "macchina di Turing" ha costituito un polo di riferimento universale.

Oggi accanto alla considerazione di un singolo automa è importante estendere lo studio alle cosiddette "reti di automi", cioè alla possibilità di collegare tra loro più automi e progettare programmi esecutivi che attribuiscono a ciascuno di essi compiti da svolgere in maniera parallela, ma ben coordinata, al fine di ottenere il risultato previsto in maniera più rapida e valorizzando una potenza di computazione distribuita. Ciò vale anche per un organizzazione di basi di dati distribuita tra più computer.

Emerge così una volta di più l'importanza di un approccio adeguato al concetto di sistema, applicato sia nei riguardi di un singolo automa, considerato come un sistema di parti interagenti tra di loro, sia alle basi di dati, viste come sistemi di organizzazione delle informazioni strutturate in dati, sia come reti di computer. Progettare un buon sistema diventa quindi una delle competenze fondamentali che entrano in gioco nell'ambito del pensiero computazionale.

5. Per progettare un percorso di formazione al pensiero computazionale

Nel 1979 formulando un primo bilancio dell'attività di formazione in informatica nell'ambito dell'Istruzione e Formazione Professionale avviato tre anni prima si erano individuati alcuni obiettivi formativi minimi da conseguire per tutti.[7] Essi erano stati articolati secondo tre ambiti: conoscenze concettuali, conoscenze e abilità procedurali, atteggiamenti.

Conoscenze di natura concettuale: concetto di algoritmo o di procedimento effettivo; concetto di automa; concetto di base di dati o di sistema informativo; concetto di linguaggio formalizzato, concetto di programma.

Conoscenze di natura procedurale (abilità): analizzare e risolvere un problema dal punto di vista algoritmico; costruire un archivio informativo o sistema infor-

[7] PELLEREY M., *L'informatica come dimensione fondamentale della Formazione Professionale di base,* in Orientamenti Pedagogici, 1979 (XXVI), 4, pp. 700-715.

mativo; tradurre un algoritmo in un linguaggio artificiale; correggere un programma in maniera sistematica.

Atteggiamenti: disponibilità a studiare situazioni o problemi in termini informatici utilizzando i concetti e procedimenti propri dell'informatica.

Come accennato nel precedente contributo, in seguito l'accento è stato spostato dall'introduzione al pensiero informatico o computazionale a un uso produttivo e sistematico degli strumenti digitali nei processi di apprendimento. Negli ultimi anni la consapevolezza che il pensiero computazionale ha, o dovrebbe avere, un ruolo importante nel processo formativo ha portato a formulare proposte di percorsi o curricoli pedagogico-didattici, anche di lungo termine. A esempio nell'agosto del 2017 è stata presentata la "Proposta di Indicazioni Nazionali per l'Informatica nella Scuola" a cura del Gruppo di Lavoro CINI[8] su "Informatica e Scuola". Si tratta di un ambizioso progetto di inserimento dell'insegnamento dell'informatica a partire dalla Scuola primaria. Per ogni ciclo scolastico ne vengono indicati sia i Traguardi da conseguire, sia gli Obiettivi specifici di apprendimento. Ai nostri fini è utile riportare i Traguardi indicati per il termine del Biennio della scuola secondaria di secondo grado.

Lo studente:

- capisce la necessità di far riferimento ad un esecutore automatico per poter esprimere algoritmi in modo non ambiguo;
- riconosce che un algoritmo risolve un problema nella sua generalità;
- giustifica la correttezza di un algoritmo rispetto a tale generalità;
- capisce la natura dei problemi che ha senso affrontare algoritmicamente;
- valuta in modo elementare l'efficienza di un programma;
- definisce, realizza e valida programmi e sistemi che modellano o simulano sistemi fisici o processi familiari del mondo reale o oggetto di studio nelle altre discipline;
- capisce quando la programmazione può costituire una soluzione vantaggiosa;
- capisce la convenzionalità della rappresentazione scelta per i dati in relazione all'informazione descritta;
- riconosce come il modo di rappresentare e organizzare i dati influisca sull'efficacia e l'efficienza nella loro elaborazione;
- sceglie e riconosce nei programmi la rappresentazione dei dati dei problemi, dei risultati che ottiene e degli elementi utili a tener traccia degli stadi intermedi dell'elaborazione;
- riconosce la natura universale e polivalente degli strumenti informatici e comprende il ruolo dei programmi nel trasformarli in macchine con finalità specifiche o particolari;

[8] Consorzio Interuniversitario Nazionale per l'Informatica.

- comprende l'importanza delle esigenze dell'utente per la realizzazione delle applicazioni informatiche;
- si rende conto che sistemi informatici, Internet e dispositivi digitali influenzano l'economia e l'organizzazione della società;
- si rende conto delle conseguenze in ambito etico e sociale della diffusione e dell'uso della tecnologia informatica ed impara a valutarla con spirito critico;
- seleziona, usa e combina programmi e servizi software per sviluppare progetti con un'articolata struttura informatica;
- seleziona, combina ed estende produzioni informatiche per esprimere la propria creatività.

Quanto agli obiettivi di apprendimento, questi vengono articolati secondo cinque ambiti: ambito algoritmi, ambito programmazione, ambito dati e informazione, ambito consapevolezza digitale, ambito creatività digitale.

Su *Mondo digitale*, la rivista dell'Aica[9], M. Lodi, S. Martini e E. Nardelli hanno proposto di identificare la meta da conseguire nel processo formativo al pensiero computazione nella formula "pensare come un informatico", di cui hanno poi descritto quattro categorie che contraddistinguerebbero tale modo di pensare: processi mentali, metodi, pratiche, competenze trasversali.[10]

Processi mentali: strategie mentali utili per risolvere problemi.

- Pensiero algoritmico: usare il pensiero algoritmico per progettare una sequenza ordinata di passi (istruzioni) per risolvere un problema, ottenere un risultato o portare a termine un compito;
- Pensiero logico: usare la logica e il ragionamento per convincersi di qualcosa, stabilire e controllare fatti;
- Scomposizione di problemi: dividere un problema complesso in semplici sotto-problemi, risolubili in modo più semplice; modularizzare; usare il ragionamento composizionale;
- Astrazione: liberarsi dei dettagli inutili per concentrarsi sulle informazioni / idee rilevanti;
- Riconoscimento di pattern: individuare regolarità/schemi ricorrenti nei dati e nei problemi;
- Generalizzazione: usare le regolarità riconosciute per fare previsioni o per risolvere problemi più generali.

Metodi: approcci operativi utilizzati dagli informatici.

- Automazione: automatizzare soluzioni; usare un computer o una macchina per eseguire compiti ripetitivi o noiosi;

[9] Associazione Italiana Calcolo Automatico.

[10] LODI M. – MARTINI S. – NARDELLI E., *Abbiamo davvero bisogno del pensiero computazionale?,* in Mondo digitale, 2017, Novembre, pp. 1- 15.

- Raccolta, analisi e rappresentazione dei dati: raccogliere informazioni e dati, interpretarli trovando schemi ricorrenti, rappresentarli in maniera appropriata; memorizzare, recuperare e aggiornare dati;
- Parallelizzazione: eseguire compiti simultaneamente per raggiungere un obiettivo comune, pensare "in parallelo";
- Simulazione: rappresentare dati e processi (del mondo reale) tramite modelli; eseguire esperimenti su tali modelli;
- Valutazione: analizzare le soluzioni implementate per giudicarne la bontà, in particolare per ciò che riguarda la loro effettività e la loro efficienza in termini di tempo impiegato o di spazio occupato;
- Programmazione: usare alcuni concetti di base della programmazione (cicli, eventi, istruzioni condizionali, operatori logici...)

Pratiche: usate tipicamente nell'implementazione di soluzioni informatiche.

- Sperimentare, iterare, fare "tinkering": nelle metodologie di sviluppo software incrementali e iterative, un progetto viene sviluppato attraverso ripetizioni di un ciclo "progetta-costruisci-verifica", costruendo in modo incrementale il risultato finale; fare "tinkering" significa costruire qualcosa usando un processo per prove ed errori, imparare dal gioco, dall'esplorazione e dalla sperimentazione;
- Testare e correggere gli errori (debug): verificare che le soluzioni funzionino provandole concretamente; trovare e risolvere i problemi (i "bug") in una soluzione o in un programma;
- Riuso e remix: costruire la propria soluzione basandosi su / utilizzando anche codice, progetti o idee già esistenti.

Competenze trasversali: modi di vedere e operare nel mondo; utili competenze per la vita favorite dal "pensare come un informatico".

- Creare: progettare e costruire artefatti, usare la computazione per essere creativi ed esprimere se stessi;
- Comunicare e collaborare: connettersi con gli altri e lavorare insieme con un obiettivo comune per creare qualcosa e per ottenere una soluzione migliore;
- Riflettere, imparare, fare meta-cognizione: usare l'informatica per riflettere e comprendere gli aspetti computazionali del mondo;
- Tollerare l'ambiguità: avere a che fare con problemi reali, aperti e non totalmente specificati a priori;
- Perseverare quando si ha a che fare con problemi difficili: essere a proprio agio nel lavorare con problemi difficili/complessi, essere determinati, resilienti e tenaci.

A parte le differenze di approccio alla definizione degli obiettivi formativi da conseguire nel promuovere il pensiero computazionale, tali descrizioni possono aiutare nel progettare itinerari formativi relativi ai vari contesti educativi.

6. Conclusione

Il 23 maggio 2018 è stata resa pubblica la nuova Raccomandazione adottata il giorno prima dal Consiglio europeo relativa alle competenze chiave per l'apprendimento permanente. Viene delineato un quadro aggiornato di quanto proposto nel 2006. Tra le otto competenze chiave considerate una riguarda l'ambito della tecnologie digitali. Sia nel 2006, sia nel 2018, viene proposta come competenza chiave la "competenza digitale", tuttavia la sua descrizione evidenzia almeno in parte la nuova consapevolezza dell'importanza di considerare quello che oggi viene definito "pensiero computazionale". Di esso nel 2006 non si ha nessun accenno. Si insiste, invece, sul "saper utilizzare con dimestichezza e spirito critico le tecnologie della società dell'informazione per il lavoro, il tempo libero e la comunicazione", e su abilità connesse con l'uso del computer "per reperire, valutare, conservare, produrre, presentare e scambiare informazioni nonché per comunicare e partecipare a reti collaborative tramite Internet".

Nel testo del 2018 appaiono alcune espressioni che evocano aspetti propri del pensiero computazionale. Si parla, infatti, di conoscenze e abilità legate "all'alfabetizzazione informatica e digitale, la comunicazione e la collaborazione, l'alfabetizzazione mediatica, la creazione di contenuti digitali (inclusa la programmazione), la sicurezza (compreso l'essere a proprio agio nel mondo digitale e possedere competenze relative alla cibersicurezza), le questioni legate alla proprietà intellettuale, la risoluzione di problemi e il pensiero critico". Tra le abilità citate si evoca: "la capacità di utilizzare, accedere a, filtrare, valutare, creare, programmare e condividere contenuti digitali".

Certo, rimane fondamentale nei processi educativi sviluppare quanto evocato nel 2006, cioè l'uso consapevole, critico e produttivo delle varie tecnologie digitali, ma ciò che non è ancora sufficientemente percepito nella sua importanza formativa è l'apporto cognitivo che può derivare ad un'introduzione sistematica e progressiva di quelle conoscenze e competenze che permettono di progettare, realizzare e verificare programmi informatici o almeno di adattare quelli disponibili alle proprie esigenze. Si è cercato in queste pagine di evidenziare alcune delle ricadute educative che possono essere conseguite. Più in generale si tratta di far almeno intuire che gli aspetti fondamentali del pensiero computazionale possono mettere in grado di non essere solo fruitori passivi delle tecnologie digitali, ma saper anche progettare e realizzare programmi più o meno complessi e impegnativi, che le macchine siano poi in grado di eseguire validamente ed efficacemente. Era questo già un obiettivo educativo presente negli Anni Ottanta dell'altro secolo, un obiettivo che poteva essere conseguito sufficientemente anche nel corso della scuola di base.[11]

[11] PELLEREY M. (a cura di), *L'informatica nella scuola media: come e perché*, Torino, SEI, 1989.

Tra le varie iniziative diffuse e a carattere internazionale che vanno in questa direzione si può citare il movimento *Maker*, che attraverso i cosiddetti laboratori FabLab promuove lo sviluppo di forme di artigianato digitale di tipo collaborativo. Si tratta, infatti, di laboratori, nei quali possono partecipare giovani e adulti, più o meno competenti, che sono attrezzati con macchine utili per la produzione di artefatti digitali, attivando processi in grado di trasformare idee in prototipi e prodotti. Un buon FabLab è un luogo di incontro tra persone con formazioni eterogenee, che risultano complementari per concepire progetti innovativi: artigiani tradizionali, esperti di elettronica, grafici, informatici. Si tratta di progettare un'app, un gioco, un archivio, valorizzando in questa impresa le varie dimensioni del pensiero computazionale alle quali abbiamo fatto riferimento. Vari di questi laboratori utilizzano una piattaforma informatica denominata *Arduino* e progettata e realizzata in Italia a Ivrea, assai utile per realizzare molteplici artefatti digitali. Nelle scuole si sta diffondendo quella che viene denominata "robotica educativa", cioè l'uso di piccoli robot (qualche volta costruiti dagli stessi studenti), che esigono, per eseguire i movimenti attesi, di essere adeguatamente programmati.

Lo sviluppo delle competenze informatiche richieste dalla programmazione è favorito inoltre dalla disponibilità di linguaggi di facile apprendimento come il diffusissimo *Phython*. Così a livello elementare sono disponibili varie piattaforme che includono la possibilità di utilizzare forme di *coding* assai semplici in gran parte derivanti dall'impianto *Logo* sviluppato a suo tempo da Seymour Papert. Tuttavia, al cuore di tutte queste possibilità di elaborazione di programmi di natura informatica sta l'utilizzazione di forme più o meno sviluppate di pensiero computazionale necessarie per progettare quanto l'esecutore automatico dovrà essere in grado di portare a termine. Viene così confermata l'affermazione di Jannette Wing circa le radici del pensiero computazionale: da una parte esso si basa sul pensiero matematico coinvolto nella soluzione di problemi di natura procedurale e, dall'altra, deve tener conto del pensiero ingegneristico, che obbliga a considerare l'impianto fisico che farà da supporto all'attuazione di quanto trovato come soluzione operativa.[12]

[12] WING J., *Computational thinking*, in Communications of the ACM, March 2006, 49, 3, p.35.