Coding, lingue e letteratura

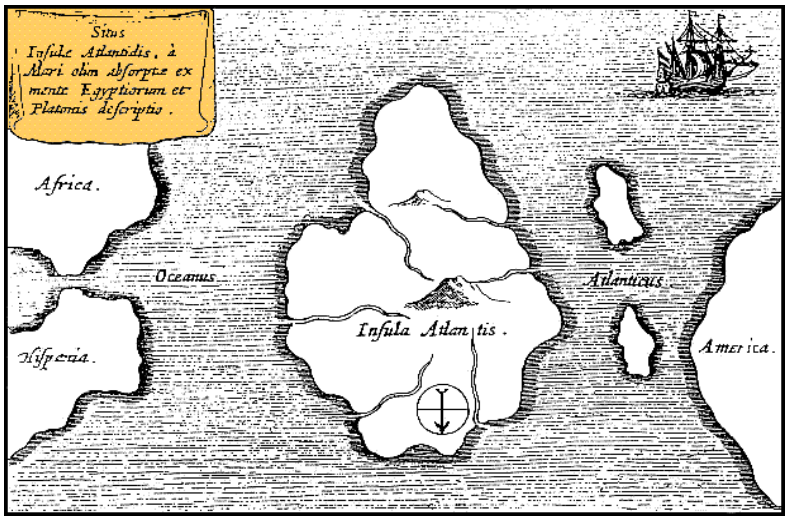
di Stefano Penge

Ivrea/Torino, 8-9 Marzo 2018

Indice generale

PROLOGO....2
Parte prima: le questioni di fondo....3
1. Linguaggi artificiali e naturali....3
2. Scrittura analogica e digitale....4
3. Schermi e tastiere....4
4. Testi digitali....5
5. E il codice sorgente?....5
Parte seconda: coding e lingua/letteratura....8
6. Coding e sperimentazione....8
7. Lingua, coding e creatività....9
Parte terza: conoscere il codice sorgente....12
8. Informatica umanistica al contrario....12
9. Codexpo.org....13

PROLOGO



Immaginate di navigare verso ovest e di incontrare un continente sconosciuto, con montagne, fiumi, strade, città.

Al centro di ogni città una biblioteca, che raccoglie i testi scritti dagli abitanti. Purtroppo sono scritti in una lingua – o forse più lingue - a voi sconosciute.

Intuite che sono testi di epoche diverse, scritte da autori diversi, che parlano di argomenti diversi.

La maggior parte sembrano testi tecnici, altri scientifici. Ma ci sono rari casi di quelle che sembrano poesie o giochi.

Questa è la situazione che ci si presenta quando ci accorgiamo dell'esistenza del continente del codice sorgente.

Un intero continente, sommerso come Atlantide.

Sconosciuto, poco studiato, a mala pena disegnato sulle carte.

Nel seguito, procederemo proprio come esploratori, avvicinandoci ai concetti di lingua naturale e artificiale, di scrittura analogica e digitale, e infine di codice sorgente.

Parte prima: le questioni di fondo

1. Linguaggi artificiali e naturali

Tutti conosciamo le lingue naturali (almeno alcune, di solito le più vicine). Ne conosciamo sintassi, lessico, esempi d'uso. Sappiamo che servono molti scopi diversi, che oltre al significato hanno una materia fonica.

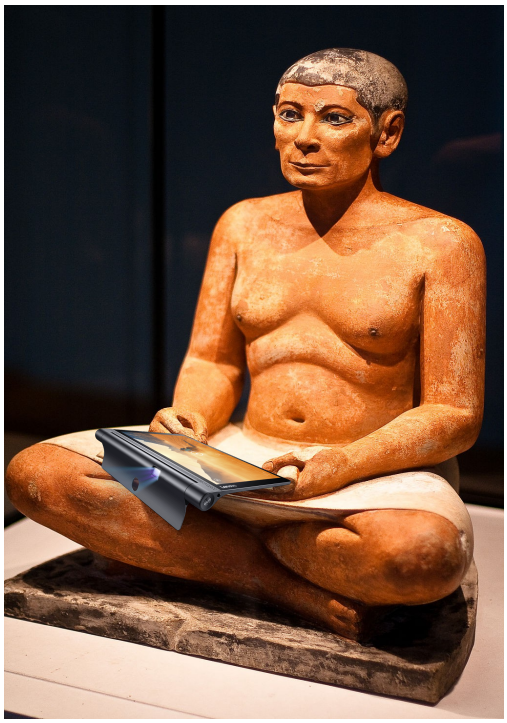
Le lingue naturali non sappiamo bene come nascono, delle più recenti possiamo seguire almeno i primi passi e l'evoluzione a partire da altre lingue.

Delle lingue artificiali (o “pianificate”) invece sappiamo tutto: chi le ha create, quando, perché.

Dalle lingue inventate (artistiche: neolingua/1984, klingon/Star Trek) a quelle vere (ausiliarie: Esperanto), dalle lingue scherzose (Europanto) a quelle digitali (html, xml, codici sorgenti)

Abbiamo l'idea che siano più limitate, meno ricche. Però ci si fanno tante cose. Per esempio, si traducono testi dalle lingue naturali a quelle artificiali, oppure si raccontano storielle, o si scrivono poesie. Anche con le lingue digitali.

2. Scrittura analogica e digitale



Chi dice lingua, parola, competenza linguistica, spesso pensa alla scrittura, come evoluzione (non solo fissazione) della parola parlata, del linguaggio orale.

Cos'è la scrittura?

- una tecnica di tracciamento di segni (con uno scalpello, un bastoncino, una penna d'uccello, una cannuccia di plastica con una sfera alla fine) su una superficie piatta e liscia (pietra, argilla, papiro, carta, schermo)?
- oppure l'intero processo di ideazione, produzione, conservazione e distribuzione di un testo?

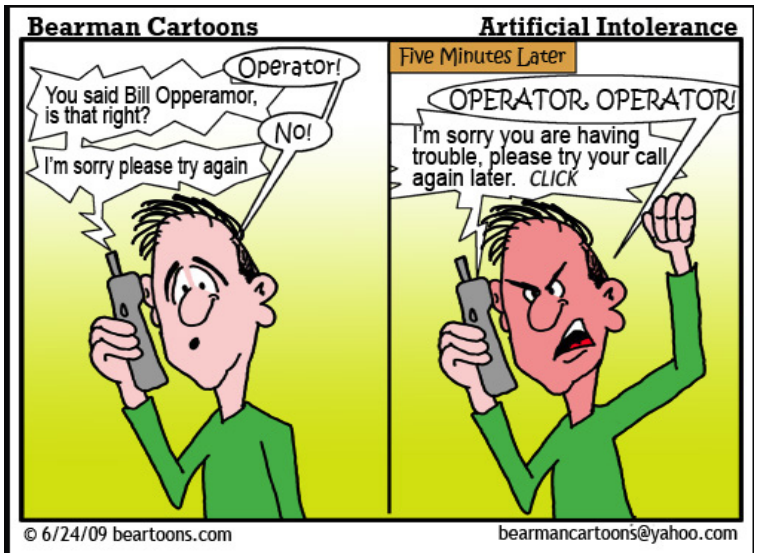
La prima concezione è quella che si ha in mente quando si contrappone la scrittura analogica a quella digitale, quando si parla di "importanza del corsivo"

La secondo è quella che si ha presente quando si parla di "cultura alfabetica". Ciò che si impara a scuola non è solo a tracciare e riconoscere segni, ma anche a far propria questa cultura: dalla produzione di testi appropriati ad un contesto o un'esigenza, alla ricerca di informazioni, alla loro trasmissione in forma di testo.

3. Schermi e tastiere

Nella prima interpretazione, si identifica la scrittura con quello che accade mettendo in relazione diretta i dispositivi di input e output, la penna e la carta, la tastiera e lo schermo.

Però: da un lato, le interfacce della scrittura sono molte di più. Dal lato dell'input: mouse, touch screen, penne digitali, fino al riconoscimento vocale; dal lato dell' output: sintesi vocale, stampante 3d (Braille), ...



Dall'altro, l'uso delle interfacce è soltanto un pezzetto del processo di scrittura, la sua forma esteriore. Quando si dice che "un computer non può scrivere poesie", non si pensa alle interfacce, ma all'ideazione, alla trasformazione, alla edizione

4. Testi digitali

```
93      <meta name="robots" content="noarchive"  data-dynamic="true">
94      <meta name="googlebot" content="noarchive"  data-dynamic="true">
95      <meta name="title" content="6 24 09 Bearman Cartoon Artificial Intelligence copy"  data-dynamic="true">
96
97      <meta name="keywords" content="phone, humor, bearman, voicerecognition, operator, artificialintelligence, bearmancartoons, comic
98
99      <meta property="og:title" content="6 24 09 Bearman Cartoon Artificial Intelligence copy"  data-dynamic="true">
100
101     <meta property="og:description" content="This cartoon by Bearman was created for Comic Strip Improv on the subject of Artificial
102
103     <meta property="og:type" content="article"  data-dynamic="true">
104
105     <meta property="og:url" content="https://www.flickr.com/photos/bearmancartoons/3657928710/"  data-dynamic="true">
106
107     <meta property="og:image" content="https://c2.staticflickr.com/4/3300/3657928710_e26e39dfc2.jpg"  data-dynamic="true">
108
109     <meta property="og:image:width" content="500"  data-dynamic="true">
110
111     <meta property="og:image:height" content="411"  data-dynamic="true">
112
113     <meta property="flickr_photos:by" content="https://www.flickr.com/people/bearmancartoons/"  data-dynamic="true">
114
115     <meta property="twitter:card" content="photo"  data-dynamic="true">
116
117     <meta property="twitter:description" content="This cartoon by Bearman was created for Comic Strip Improv on the subject of Artif
118
119     <meta property="twitter:app:url:iphone" content="flickr://flickr.com/photos/bearmancartoons/3657928710/"  data-dynamic="true">
120
121     <meta property="al:ios:url" content="flickr://flickr.com/photos/bearmancartoons/3657928710"  data-dynamic="true">
```

La differenza sostanziale che introduce il digitale è legata alla natura del testo (indipendentemente dalle interfacce): infinitamente modificabile, traducibile in altri linguaggi/media, replicabile…

Questa natura speciale (che non oppone linguaggi naturali e artificiali, ma anzi permette di creare versioni digitali di qualsiasi testo) è quella che rende possibile l'informatica umanistica (Digital Humanities): applicare strumenti e concetti informatici ai testi: contare le occorrenze, cercare le somiglianze, oppure rendere espliciti i significati tramite dei "marcatori" (OpenGraph) per permettere la nascita del "web semantico", quell’universo di testi in scritti in modo che sia facile, per i programmi, categorizzarli in base ai contenuti, e in qualche modo capirli.

Ma non necessariamente sono sempre cose serie, né sempre cose pensate per essere lette solo da automi…

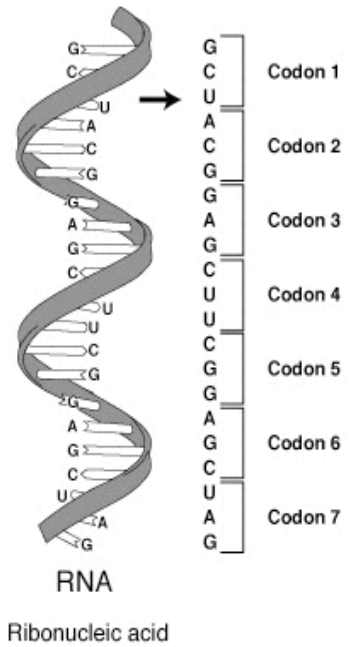
```
45      <meta property="al:ios:app_name" content="Flickr" />
46      <meta property="al:ios:app_store_id" content="328407587" /> <meta property="twitter:app:name:iphone" content="Flickr" />
47      <meta property="twitter:app:id:iphone" content="328407587" />
48      <meta property="twitter:site" content="@flickr" />
49          <script type="text/javascript" nonce="2c60e8e921fbc17eb67cf0f9e4adb594">
50          !function(e,o){e.beaconError=function(o,n,t){try{if(-1===String(e.location.host).indexOf(".flickr.com"))return}catch(
51          </script>
52          <!--
53                      . .  ` ;  `-   ,  ,  ``   .                 _ ,` `  .
54                  ' ,gi$@$q  pggq   pggq  .                  ' pggq
55                + j@@@P*\7  @@@@   @@@@                      ; @@@@ !  `_  ,  `   _`  ,  `  `
56              . .  @@@K      @@@@          ;  -  `  _ _  .  @@@@ ;/
57              ; pgg@@@@gggg  @@@@   @@@@ .' .iS@@@@@Si  @@@@  .6@@@P' !!!! j!!!7 ;
58              `  @@@@@@@@@@  @@@@   @@@@  `  j@@@P*"*+Y7  @@@@ .6@@@P  `  !!!!47*"*+;
59              `_   @@@@      @@@@   @@@@  .@@@7  .      `  @@@@.6@@@P  `  !!!!;  .  `  '
60                `  @@@@   '  @@@@   @@@@  ;@@@!  !:      @@@@7@@@K  `; !!!!  '   `  '
61                   @@@@   .  @@@@   @@@@  `%@@@.      .  @@@@ 7@@@b  . !!!!  :
62                !  @@@@      @@@@   @@@@   \@@@$+,,+4b  @@@@ `7@@@b   !!!!
63                   @@@@   :  @@@@   @@@@     7%S@@hX!P' @@@@   `7@@@b  !!!!  .
64                ;  """"      """"   """"  ;.      ^^^^`    """"   `"""""  !!!!
65                `  -  .   ,                 ,                  _`_        .  _  ,  -
66                          ,  `  .,llllllllll,     `-; '     .~ . `. `. ~.
67                            ,jlllllllllllllp.  .!'  .+. . . . . . .+.  `.
68                        ` jlllllllllllllllllll   `   +. . . . . . . .+  .
69                      .  jlllllllllllllllllll   . . . . . . . . . . .
70                      .l@@@@@@@llllllllllllll. j. . . . . . . ;;;;;;;l `
71                    ; ;@@@@@@@@@@@@@@@@@@@@@llll ;. . ;;;;;;;;;;;;;;;;;; ;
72                      ;l@@@@@@@@@@@@@@@@@@@@@@l; ;;;;;;;;;;;;;;;;;;;;;;;;
73                    `  Y@@@@@@@@@@@@@@@@@@@@@P   ;;;;;;;;;;;;;;;;;;;;;;  '
74                     -  Y@@@@@@@@@@@@@@@@@@@@P  .  ;;;;;;;;;;;;;;;;;;;;  .
75                          `*@@@@@@@@@@@@@@@@*  `  `  `;;;;;;;;;;;;;;;;;`
76                          `.   `*%@@@@@@@%*  '.        `   +;;;;;;;;;+`  '
77                                  '  .  '    -                 - '  .  `  '  .
78
79
80              You're reading. We're hiring.
81              https://flickr.com/jobs/
82
83            Don't see something there for you? Email us iwanttowork@flickr.com
84          -->
85      <title>6 24 09 Bearman Cartoon Artificial Intelligence copy | Flickr</title>
```

5. E il codice sorgente?

Tranne i programmatori, nessuno ha mai nemmeno visto un codice sorgente. Magari si immagina Matrix, oppure sequenze di 0 e 1.

Invece sono cose come questa:

```
int
foo(int k) {
    if (k < 1 || k > 2) {
        printf("out of range\n");
        printf("this function requires a value of 1 or 2\n");
    } else {
        printf("Switching\n");
        switch (k) {
        case 1:
            printf("1\n");
            break;
        case 2:
            printf("2\n");
            break;
        }
    }
}
```



E' come il codice genetico di un organismo. E' ciò che rende un programma quello che è, che definisce come si presenta e come reagisce all'ambiente.

Il codice sorgente è un tipo di testo, solo che nasce (e di solito resta) digitale. Ma a parte la maniera di scriverlo e leggerlo (tastiere, schermi o altro), e il fatto di essere scritto in un linguaggio artificiale (detto "di programmazione") che non è pensato per la comunicazione tra umani, condivide con i testi analogici molti aspetti: è scritto in un certo momento, da qualcuno, per uno scopo, con un progetto in testa. Viene letto da altri, viene modificato, diffuso, tradotto, pubblicato.

Il codice sorgente viene progettato prima di essere scritto, in una maniera che assomiglia al modo in cui si inventa una storia.

In un contesto definito ci sono dei personaggi che cercano di raggiungere un obiettivo, superando gli ostacoli che

incontrano.

A volte hanno bisogno di aiuto esterno, a volte, fanno dei piani che falliscono e devono tornare indietro.

Si potrebbero riconoscere alcune delle funzioni di Propp, che sono state applicate ai film e ai videogiochi. Ad esempio,

2. Divieto

3. Infrazione

4. Ricognizione

5. Ottenimento

10. Consenso

12. Funzione del donatore

15. Trasferimento

19. Rimozione

20. Ritorno

25. Prova

26. Superamento

27. Identificazione

31. Matrimonio o Incoronazione

Ma perché pensare al codice sorgente come un testo?

Intanto perché è un modo di acquisire una visione più generale di testi, linguaggi e autori: non concentrarsi sulle interfacce e sulla materia, ma sui processi di progettazione, realizzazione, diffusione. E questo è un guadagno anche per lo studio e per la didattica della lingua e della letteratura.

Poi perché, socialmente, questo tipo di testo ha un'importanza sempre più grande, ma sempre meno evidente, invisibile anche per chi li usa tutti i giorni. Sapere che *"gli algoritmi non sono cattivi, ma li disegnano così"* è qualcosa che riguarda tutti noi, a partire dai ragazzi.

Perché quello della scrittura di codici sorgente è un mestiere che sarebbe limitante ridurre alla sola giustapposizione di simboli astrusi.

Perché le competenze per scrivere un codice non sono solo quelle logico-ingegneristiche, ma anche quelle narrative; probabilmente vanno insegnate anche queste se si vuole una generazione di "programmatori consapevoli e creativi".

Insomma, affrontare il codice sorgente da questo punto di vista può avere un impatto positivo

sulla formazione, sia dal lato consapevolezza/cittadinanza digitale che da quello della capacità di orientarsi e scegliere una strada professionale.

Parte seconda: coding e lingua/letteratura

6. Coding e sperimentazione

Coding come gioco, come preparazione al lavoro futuro, come apprendimento del pensiero computazionale… siamo sicuri?

Qui propongo un altro punto di vista: usare il coding è un modo di costruire (un macchina che corrisponde ad) una teoria del mondo, e di verificarne il funzionamento.

Una teoria fisica, biologica, geografica, storica, ma anche linguistica o letteraria.

Ad esempio: definiamo un modello di frase (una "grammatica") e poi scriviamo il codice sorgente che sia in grado di produrre frasi casuali secondo quel modello

Oppure un programma che sia in grado di riconoscere la correttezza di una frase italiana secondo quel modello.

Questo è un esempio di grammatica scritta nel linguaggio DCG (Definite Clause Grammar). Non è solo una descrizione, ma proprio un pezzo di codice sorgente, che si può utilizzare così com'è in Prolog (che è un linguaggio di programmazione particolare, nato a Marsiglia e dedicato proprio alle elaborazioni del linguaggio naturale).

```
frase --> s_oggetto, predicato.
s_oggetto --> articolo, nome.
predicato --> verbo, avverbio, s_oggetto.
articolo --> [il].
articolo --> [un].
nome --> [chirurgo].
nome --> [coccodrillo].
verbo --> [pizzica].
avverbio -->[sempre].
avverbio -->[talvolta].
```

Una volta definito nostro modello (Soggetto + Verbo + Avverbio + Oggetto) vogliamo scrivere

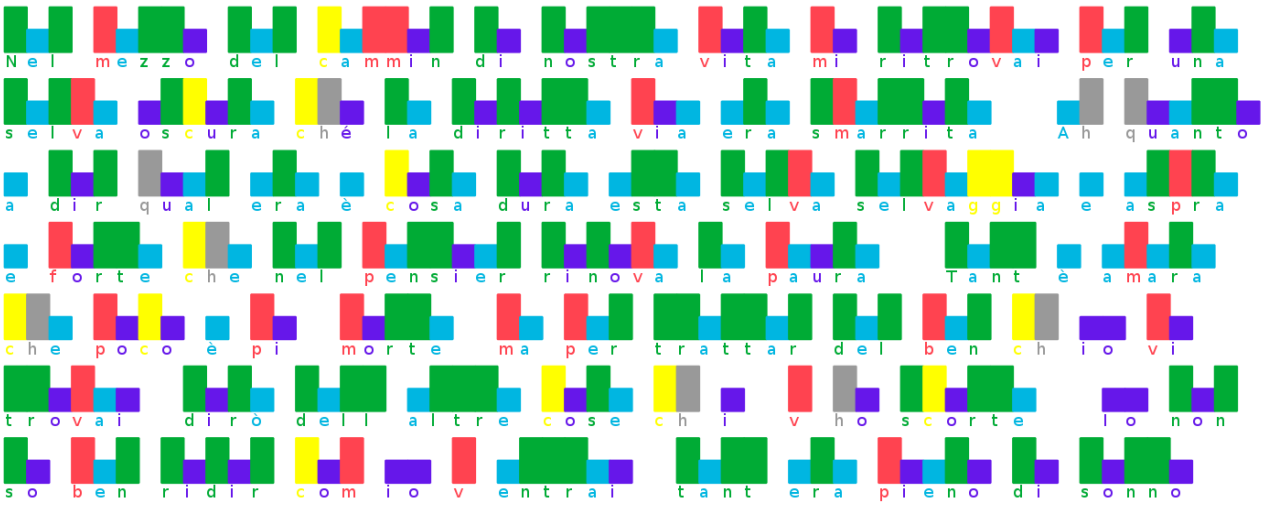
un codice sorgente che sia in grado di produrre frasi corrette, oppure che ci dica che:

"il chirurgo pizzica sempre un dinosauro" è corretta

"il pizzica chirurgo un sempre dinosauro" non è corretta

"il chirurgo elegante discute talvolta con un dinosauro verde" NON è corretta (secondo il modello: infatti non sono previsti aggettivi né preposizioni).

Un altro esempio di attività al limite tra arte, linguistica e letteratura:



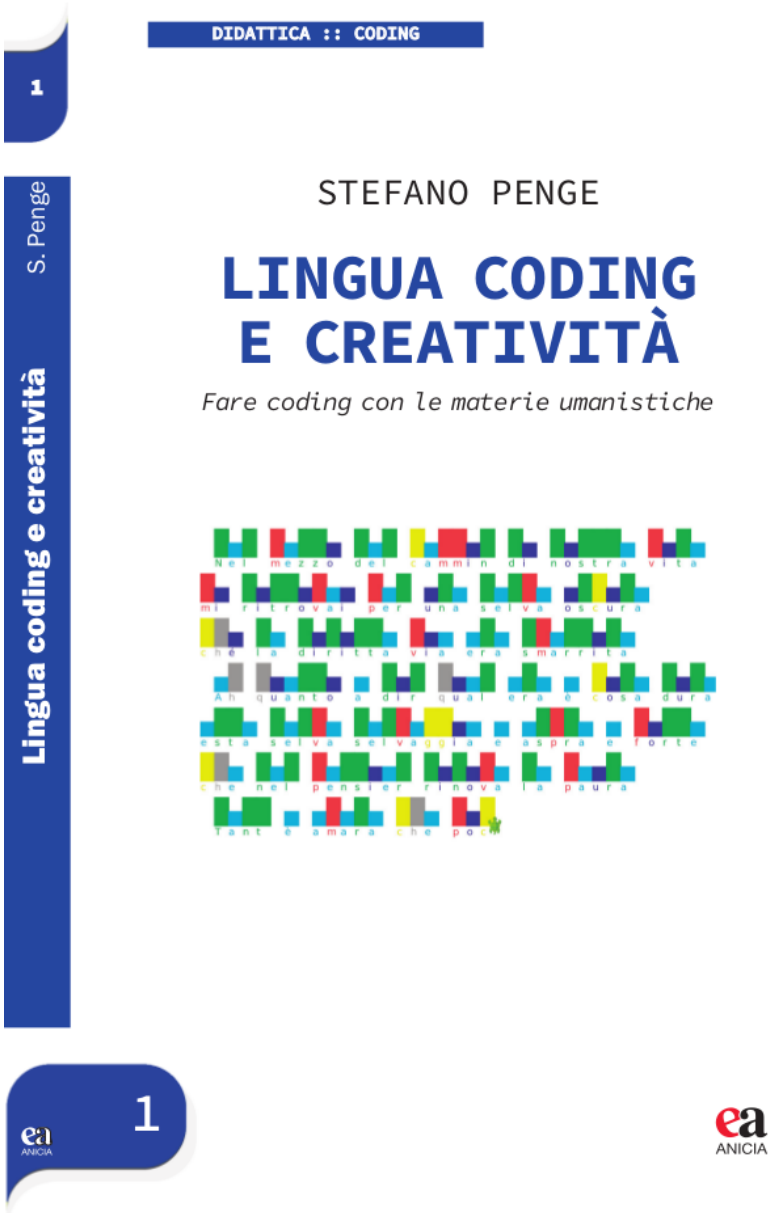
Fiumi di inchiostro sono stati scritti sull’uso della fonetica per rappresentare il tono affettivo, l’ambiente (le "rime aspre e chiocce" del canto XXXII): ma come si fa a scoprire quali gruppi di consonanti, o sequenza di consonanti e vocali, portano con sé un certo sentimento (un po’ come i modi minore e maggiore in musica)?

E se provassimo rappresentare visivamente vocali e consonanti? E a visualizzare con colori diversi i gruppi di consonanti?

Che succederebbe se confrontassimo in questo modo i canti del Paradiso con quelli dell'Inferno?

O Manzoni con D'annunzio?

7. Lingua, coding e creatività



Dopo aver insegnato (molti anni fa) il Logo a scuola, e dopo aver passato il resto del tempo a scrivere programmi per lavoro, l'anno scorso ho iniziato a occuparmi di Coding, proprio in occasione del seminario di Ivrea. Non ero troppo convinto della maniera in cui veniva proposto, della ragioni e degli ambienti e ho scritto un piccolo ebook “Dietro al coding” (che potete scaricare da qui) in cui raccontavo la storia recente e meno recente dell’introduzione della programmazione nella scuola dell’obbligo, la storia dei linguaggi didattici dal BASIC a Scratch, presentavo alcune opportunità meno conosciute in Italia. Però è facile criticare: e allora ho cominciato a immaginare un modo diverso di fare coding, ad esempio con un ambiente testuale (Kojo) inventato da un programmatore indiano, Lalit Pant. E siccome ero convinto che il senso del coding non necessariamente legate alle discipline STEM, il tema scelto è stato quello della lingua e della letteratura. Ne è nato è un libro cartaceo, che è appena uscito (marzo 2018), con un companion site (http://www.anicialab.it/coding/lingua).

E’ un testo che presenta una serie di attività didattiche di "coding" proprio intorno ad argomenti di lingua. E' probabilmente l'unico, al momento, nel suo genere, non solo in Italia. L'idea è nata dalla rilettura di un testo rivoluzionario del 1989[1] in cui Giorgio Casadei, ordinario di Informatica a Bologna, presentava un percorso di apprendimento del Prolog dedicato a insegnanti di materie umanistiche. Se si poteva pensare di usare un linguaggio di programmazione (e uno davvero particolare come il Prolog) in una classe di Italiano trent'anni fa, perché non oggi che la programmazione è tornata in auge in modo così potente?

1 G.Casadei, P.Cuppini, A. Palareti, Informatica per le discipline umanistiche. Applicazioni didattiche del Prolog, Zanichelli, 1989.

La domanda che probabilmente si porrà il lettore è: ma perché è stato scelto proprio questo dominio, così lontano da computer e programmi? Nella lingua non ci sono numeri, non ci sono grandezze fisiche: cosa ci sarà mai da calcolare? E perché andare a disturbare i docenti di Italiano e Latino, che non sentono nessun bisogno di introdurre il coding nelle loro classi?

I motivi di questa scelta sono in realtà più di uno: prima di tutto, si tratta di una sfida. L'intento è quello di offrire un esempio di esplorazione di campi diversi da quello scientifico, campi in cui generalmente i docenti non hanno grandi competenze informatiche (e proprio per questo hanno bisogno di maggiori stimoli). Una mancanza di competenze derivata dalla separazione netta tra area umanistica e area scientifica che ha radici storiche profonde: la divisione medievale delle arti liberali in trivio (grammatica, retorica e dialettica) e quadrivio (aritmetica, geometria, astronomia e musica), la definizione di curricoli scolastici e universitari; ma, a mio avviso, deriva anche da una generale incomprensione della natura dell'informatica, intesa come tecnica che tratta numeri (in fondo "computer" significa "calcolatore", no?).

Informatica però significa "trattamento automatico dell'informazione", e informazione non equivale solo a "quantità numeriche". La linguistica computazionale – che si occupa di testi e letteratura – è una branca dell'informatica (o meglio, una disciplina all'incrocio di altre discipline) che ormai esiste da anni e ha una sua dignità che non deve essere più dimostrata. Esiste addirittura un'informatica umanistica, che nel mondo prende il nome di "digital humanities". Meno frequente, ma altrettanto produttiva, è la direzione inversa, cioè l'applicazione di concetti e metodi delle discipline umanistiche (come la linguistica) all'informatica. In fondo esistono circa duemila linguaggi di programmazione e una biblioteca sterminata di testi scritti: possibile che a nessuno venga in mente di studiarli come tali?

Il secondo motivo è legato quindi al fatto banale che se nella lingua non ci sono numeri questo non significa che non si possa utilizzare un computer in maniera interessante. Anzi: non solo si può, ma per certi versi è ancora più sensato. Se per lavorare in classe con la fisica e un computer o si usano sensori e attuatori, oppure si entra nel campo delle simulazioni (con i vantaggi e i limiti di questo approccio didattico), per lavorare con la lingua con un computer non serve niente altro. I computer sono tradizionalmente ben equipaggiati per trattare la lingua, soprattutto quella scritta, anche se oggi se la cavano bene anche con il parlato. Sanno produrre stringhe di caratteri, frasi e persino testi veri, non solo *simulazioni* di testi. Il prodotto di un automa linguistico non è un disegno o un'animazione che va interpretata come schema di un fenomeno fisico: *è il fenomeno stesso.* Questa situazione, che è simile a quella della matematica, deriva dal fatto che la lingua può essere trattata come un insieme di simboli, che possono essere codificati, analizzati, combinati, letti e prodotti. Non significa naturalmente che *tutte* le competenze linguistiche possano essere apprese in questo modo; e infatti l'oggetto delle proposte che trovate più avanti è spesso all'interno del sottoinsieme della grammatica, intesa come insieme delle regole che governano la produzione di enunciati, o più in generale di sequenze di simboli.

Ad esempio, nell'attività "E perché non un'automa?"[2], l'obiettivo è costruire insieme un *automa linguistico*, cioè un programma in grado di simulare un parlante della lingua Italiana, in una situazione specifica molto semplice: premettere l'articolo ad un nome.

2 Si, l'apostrofo è voluto. Si capirà perché durante la lettura

Perché proprio gli articoli? Perché in Italiano sono abbastanza semplici (almeno a prima vista), ma non così semplici come, ad esempio, in Inglese. Perché si prestano a discutere di regole, di eccezioni, di ambiguità; permettono di andare a toccare la storia della lingua e le parentele tra lingue romanze; hanno aspetti fonetici e aspetti semantici; eccetera.

In realtà l'oggetto è abbastanza indifferente, perché non si tratta di un'attività per insegnare *cosa sono gli articoli e come si usano*. Costruire un automa in grado di assegnare la forma giusta di articolo ad un nome è soprattutto un modo di *riflettere sulle competenze linguistiche apprese*, di scomporle, di rappresentarle in modo da essere in grado di applicarle anche a casi particolari e poi di generalizzarle.

Nel libro sono proposte 10 attività, o meglio dieci modelli, da praticare in classe. Ad esempio:

- come si accordano gli articoli ai nomi

- come si coniuga un verbo a partire dalla radice

- come si misura la leggibilità di un testo

ma anche:

- come si inventa una lingua

- come si impara a vincere al gioco Sasso, Forbici e Carta

- come si trasformano le frasi musicali in animazioni

Webografia, esempi di codice sorgente e tutorial sono liberamente accessibili dal companion site del libro: http://www.anicialab.it/coding/lingua .

Probabilmente i docenti di materie umanistiche che leggeranno questo testo troveranno imprecisioni, errori, mancanze. Mea culpa: non sono un linguista, né un docente di lingua. Tutte le attività si possono senz'altro correggere e migliorare, o ripensare da zero. In generale, quanto scritto rappresenta solo un esempio di *una possibile modalità* di utilizzo della programmazione come strumento di costruzione collettiva di conoscenze, da adattare (o ripensare) nei contesti specifici della classe.

Infine: il libro è indirizzato ai docenti di lingua e letteratura, ma spero possa essere interessante e utile anche per tutti gli altri.

Parte terza: conoscere il codice sorgente

8. Informatica umanistica al contrario

Se l’informatica umanistica applica gli strumenti digitali ai testi analogici, perché non dovrebbe essere possibile fare il contrario?

Come dicevamo più in alto: se i codici sorgenti dei programmi sono testi, allora possono essere studiati con strumenti e concetti delle scienze umanistiche: linguistica, sociolinguistica, stilistica, ...

Se il contesto (culturale/storico) influenza la produzione di testi musicali, poetici, etc, perché gli artefatti linguistici digitali dovrebbero sfuggire a questa regola?

Se esistono configurazioni di tratti minimi riconoscibili che sono comuni a gruppi di autori e di testi, potrebbero essere rintracciati anche per i codici sorgenti?

Insomma occorre porsi delle domande:

Quanti e quali sono i linguaggi di programmazione? perché vengono inventati, e da chi, e dove?

Chi scrive codice? Età, genere, cultura, lingua, ...

Come lo scrive? A casa, in garage, in ufficio, ...

Si possono scrivere codici belli o brutti, oltre che funzionanti o sbagliati?

C'è spazio per una dimensione artistica? E quanto è collegata con la valutazione del lavoro dei programmatori?

Di chi sono i codici sorgenti? C’è spazio per una riflessione politica?

Quali risultati ci si aspetta di ottenere ? A che serve porsi domande come queste?

Per certi versi, è la domanda ad essere strana. Nessuno si domanda a cosa serve studiare la storia della composizione musicale, o la storia delle scuole pittoriche occidentali. Nessuno si domanda a cosa serve lo studio della nascita dello stile manzoniano condotta attraverso l'analisi dei brogli dei Promessi Sposi. Il codice sorgente è una parte della nostra cultura, a livello planetario. *“Homo sum: nihil humano a me alienum puto”*.

Ci possono però essere motivazioni pratiche, legate alla necessità di rendere trasparente il mondo del software in cui siamo immersi; alla opportunità lavorative che vanno sprecate; al miglioramento della qualità del lavoro.

Comunque, a queste domande non è stata mai data una risposta in un quadro unitaria. Persino nelle facoltà di informatica l’aspetto storico, sociologico, linguistico non viene preso in considerazione.

A dire il vero, ci sono stati dei precursori illuminati (Pierre Lévy), e cominciano a nascere oggi

dei laboratori di ricerca e formazione (come quello di *Critical Code Studies*, Mark Marino, Università del Sud della California a Los Angeles).

Ma si resta nella nicchia degli esperti. Perché invece non presentare a tutti il mondo del codice sorgente? E perché non con una mostra?

9. Codexpo.org

L'idea di una mostra mi è venuta almeno dieci anni fa, mentre insegnavo in un corso di laurea magistrale a Scienze della Comunicazione. All'inizio ho avuto un po' di difficoltà a spiegarne il senso, tanto agli informatici che agli umanisti. Poi finalmente nel 2017 sono riuscito a raccogliere intorno all'idea della mostra un manipolo di coraggiosi: non solo programmatori, ma anche docenti.

Abbiamo creato un'Associazione di Promozione Sociale – codexpo.org - e un sito – www.codexpo.org - , e cominciamo a raccogliere segnali di interessamento da parte di altre Associazioni (come Gessetti Colorati, AICA), fondazioni (Fondazione Mondo Digitale), enti di formazione (CESEDI Torino), Università (Dipartimento di Scienze della Comunicazione della Sapienza), Reti di imprese (Rete Italiana Open Source), imprese (Digizen).

Codexpo è la prima mostra dedicata al Codice Sorgente, quel testo magico che sta dentro a computer e cellulari, controlla i satelliti e fa funzionare i siti web, come questo.

Una mostra che per la prima volta raccoglie e presenta la vera anima dei computer: il codice sorgente.

Proprio per questa centralità del codice sorgente e nella sua forma originaria, Codexpo è fortemente legato al concetto di Open Source come condizione necessaria: leggere il codice significa poterlo verificare, modificare, imparare dagli esempi e poterlo condividere. Per ricordare questo aspetto, tutti gli strumenti utilizzati sono e saranno Open Source.

Potremmo anche chiamarlo "Mostra di letteratura digitale". Il termine è qui usato in maniera provocatoria in tempi di e-book: non la letteratura tradizionale resa digitale, né la letteratura scritta da software intelligenti o considerati tali, ma piuttosto il campo infinito di testi scritti da umani negli innumerevoli linguaggi di programmazione.

Una sconfinata letteratura (che cresce ad un ritmo inimmaginabile) di cui generalmente non si intuisce nemmeno l'esistenza, di cui poco si sa e che raramente è studiata come tale. Milioni di "testi" accessibili anche grazie ai repository pubblici di codice, come per esempio Github o SourceForge, ma anche Gitlab o Bitbucket.

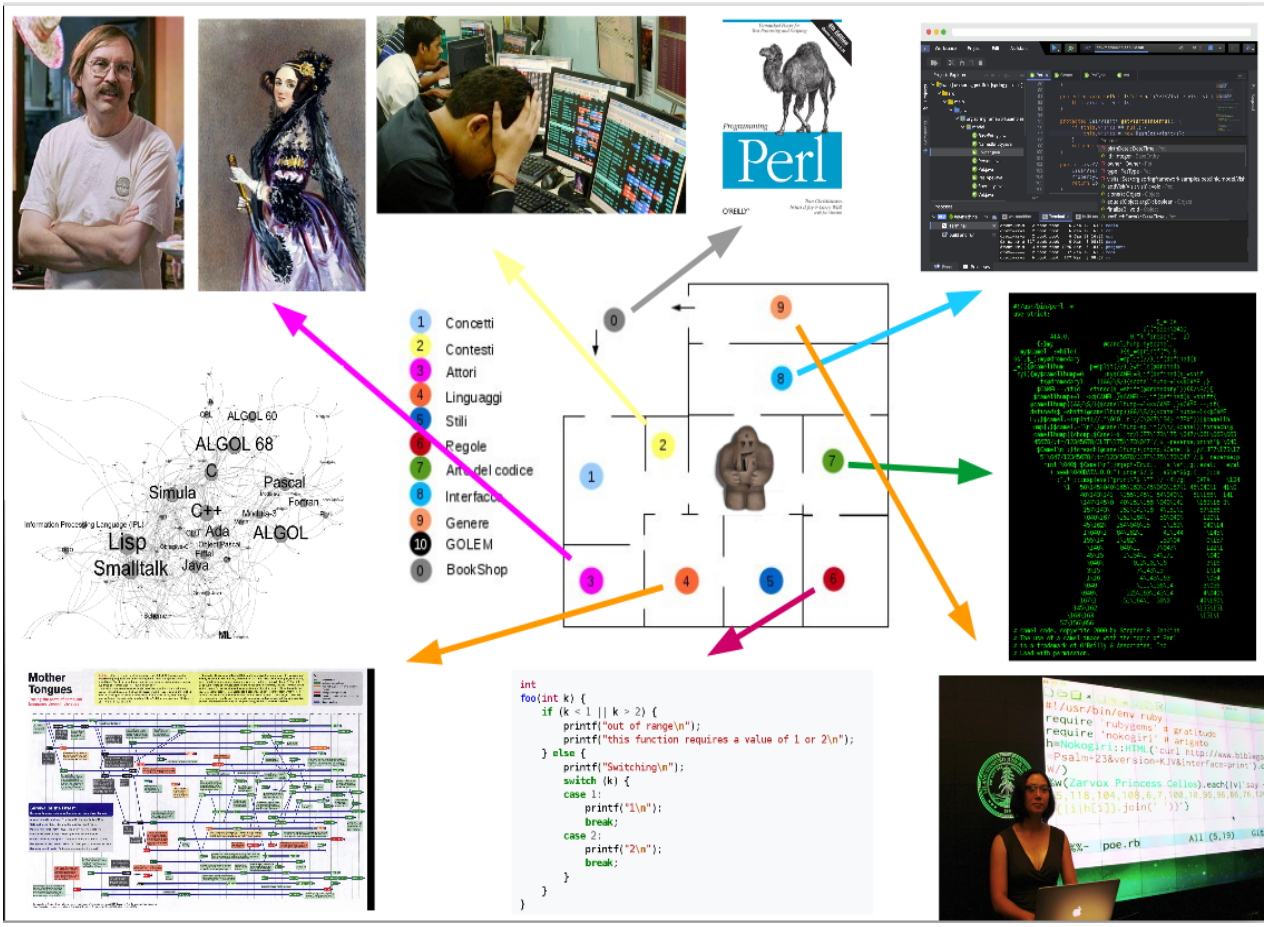
In una prospettiva che vede i codici sorgenti equiparabili in qualche modo ai testi scritti da autori in determinate lingue storiche e in determinati contesti, l'esposizione vuole fornire un'informazione che sia corretta ed attendibile, ma anche stimolante e non solo per gli addetti ai lavori.

Per questo motivo, accanto alla schede informative verranno installate opere multimediali digitali e non che punteranno, attraverso rappresentazioni sensorialmente ricche, a rimuovere il blocco che impedisce a tutti noi di pensare al codice sorgente di un programma per computer come ad un oggetto apparentato con altri testi ma di tipologie più conosciute, come sceneggiature teatrali e brani musicali.

Codexpo parte dagli *oggetti* (i programmi) ma presenta anche le *azioni* della scrittura e lettura di un testo e richiama l'attenzione sulle *persone* che scrivono, sulle loro idiosincrasie, sugli stili.

Verranno collocati uno a fianco all'altro programmatori e inventori di linguaggi, opere artistiche e programmi interattivi.

Di ogni oggetto si cercherà di restituire il contesto, storico ma anche culturale: quando si scrive, come, con quali riferimenti, dentro quali metafore.



Questi i nome delle stanze:

1. **Concetti**: codice, interpreti, compilatori
2. **Contesti**: come e dove si scrive il codice
3. **Attori**: da Leibniz a Wirth, i personaggi che hanno contribuito all'invenzione dei linguaggi
4. **Linguaggi**: oltre duemila lingue diverse, da Algol a vvvv
5. **Stili**: manuali, raccomandazioni, mode...
6. **Regole e improvvisazione**: lo spazio della creatività
7. **Arte del codice**: dall'ASCII art alle rappresentazione 3D
8. **Interfacce**: come gli umani interagiscono con i computer
9. **Coding e genere**: luoghi comuni e personaggi fuori dal comune
10. **Il GOLEM**: la macchina che si nutre di codice sorgente

Mostra che si estende e moltiplica in attività di presentazione, formazione, informazione anche al di fuori dalle sue sale. E, tenuto conto dei tempi rapidissimi di evoluzione dell'oggetto, una mostra che muta continuamente, che si arricchisce e che si ristruttura.

Codexpo.org organizza eventi di presentazione del progetto e, più in generale, di condivisione delle nostre idee ed esperienze.

Racconteremo di quanti linguaggi di programmazione esistano e perché, vi mostreremo quelli più strani (da Piet a LOLcode).

Racconteremo la storia di alcuni personaggi che sono stati fondamentali nell'invenzione dei linguaggi di programmazione, da Ada Lovelace a Nicklaus Wirth passando per Larry Wall o John McCarthy e salutando Yukihiro Matsumoto.

Presenteremo le gare di codice offuscato, gli haiku scritti in Perl e la prima poesia scritta in ALGOL, gli strumenti per scrivere codice sorgente musicale (a partire dall'Arca di Athanasius Kircher), e tanto altro.

Contatti

stefano.penge@gmail.com

info@codexpo.org