

Prefazione

Ho accettato con grande piacere l'invito rivoltomi a scrivere queste poche righe di prefazione per almeno due ragioni. La prima è che la richiesta mi viene da due colleghi che ho sempre tenuto in grandissima considerazione, fin dal tempo in cui li ho potuti conoscere ed apprezzare come studenti e come giovani ricercatori.

La seconda ragione è che il testo di Gabbrielli e Martini è molto vicino al libro che io avrei sempre voluto scrivere e, per ragioni varie, non ho mai scritto. In particolare, l'approccio seguito dal libro è quello che io stesso ho seguito nell'organizzazione di vari corsi sui linguaggi di programmazione che ho insegnato, in diverse fasi e sotto diverse etichette, per quasi trent'anni.

L'approccio, schematizzato in due parole, è quello di introdurre i concetti generali (sia dei meccanismi linguistici che delle corrispondenti strutture di implementazione) in modo indipendente da linguaggi specifici e solo in un secondo tempo collocare nello schema i "linguaggi veri". Questo è l'unico approccio che permette di mettere in evidenza le similitudini tra linguaggi (ed anche tra paradigmi) apparentemente molto diversi. Allo stesso tempo si rende così più facile il compito di imparare linguaggi diversi. Nella mia esperienza di docente, anche dopo molti anni gli ex-studenti si ricordano i principi appresi nel corso e continuano ad apprezzarne l'approccio, che ha loro permesso di adattarsi alle evoluzioni delle tecnologie senza grandi difficoltà.

Il testo di Gabbrielli e Martini ha come riferimento prevalente un corso della laurea triennale in Informatica. Per questa ragione non ha prerequisiti complessi e affronta l'argomento trovando un perfetto equilibrio tra rigore e semplicità. Particolarmente apprezzabile e riuscito è lo sforzo di mettere in luce il collegamento con alcuni "pezzi di teoria" importanti (come i linguaggi formali, la calcolabilità, la semantica), che il libro giustamente richiama, anche perché in molti corsi di laurea triennali questi argomenti non vengono più trattati.

Prima di concludere, mi permetto un accenno polemico al fatto che purtroppo in alcuni corsi di laurea triennali (tra questi, quello in cui io insegno) sono stati eliminati dai corsi fondamentali oltre ai tradizionali contenuti di informatica teorica anche i contenuti di linguaggi, considerati anch'essi troppo teorici per un corso professionalizzante orientato alle nuove tecnologie. Gli studenti conoscono (o credono di conoscere) alla perfezione un unico linguaggio, magari Java, con tutte le cose "che servono", incluse le librerie. Ma non sapranno mai descrivere il linguaggio che usano in termini di concetti generali, perché nulla sanno di metodi per il passaggio di parametri, di regole di scoping e di polimorfismo. Almeno ora avranno la possibilità di imparare queste cose su un ottimo testo scritto in italiano.

Giorgio Levi

Facilius per partes in cognitionem totius adducimur.
(Seneca, *Epist.*, 89, 1)

L'apprendimento di un linguaggio di programmazione costituisce per molti studenti il rito d'iniziazione all'informatica. Si tratta di un passaggio importante, ma che presto si rivela insufficiente: tra gli attrezzi del mestiere ci sono molti linguaggi ed una competenza importante del bravo informatico è quella di saper passare da un linguaggio all'altro (e di apprenderne di nuovi) con naturalezza e velocità.

Questa competenza non la si acquisisce soltanto imparando *ex novo* molti linguaggi diversi. Come le lingue naturali, anche i linguaggi di programmazione hanno tra loro somiglianze, analogie, fenomeni di importazione dall'uno all'altro, genealogie che ne influenzano le caratteristiche. Se è impossibile imparare bene decine di linguaggi di programmazione, è possibile conoscere a fondo i meccanismi che ispirano e guidano il progetto e l'implementazione di centinaia di linguaggi diversi. Questa conoscenza delle "parti" facilita la comprensione del "tutto" costituito da un nuovo linguaggio, fornendo dunque una competenza metodologica fondamentale nella vita professionale dell'informatico, in quanto consente di anticipare l'innovazione e di sopravvivere all'obsolescenza delle tecnologie.

È per questi motivi che un corso sugli aspetti generali dei linguaggi di programmazione è, in tutto il mondo, un passaggio chiave della formazione avanzata (universitaria o professionale) di un informatico. Relativamente ai linguaggi di programmazione, le competenze fondamentali che un informatico deve possedere sono almeno di quattro tipi:

- gli aspetti propriamente linguistici;
- come i costrutti linguistici possono essere implementati ed il relativo costo;
- gli aspetti architetturali che influenzano l'implementazione;
- le tecniche di traduzione (compilazione).

È raro che un singolo corso possa affrontare tutti e quattro questi aspetti. In particolare, la descrizione degli aspetti architetturali e delle tecniche di compilazione costituiscono ciascuno un argomento sufficientemente complesso ed elaborato da meritare una trattazione separata. Gli altri due aspetti sono il contenuto primario di un corso generale sui linguaggi di programmazione e costituiscono il contenuto principale di questo testo.

La letteratura anglosassone, a differenza di quella in lingua italiana, è ricca di testi che affrontano questi argomenti, alcuni carichi di storia e sui quali si sono

formate generazioni di studenti. Tutti questi testi, tuttavia, hanno in mente un lettore avanzato che conosca già diversi linguaggi di programmazione, che abbia una competenza non superficiale dei meccanismi di base fondamentali, che non si spaventi davanti a frammenti di codice espressi in linguaggi a lui ignoti (perché riesce a comprenderli per analogia o per differenza da quelli che già conosce). Si tratta di testi, dunque, che potremmo chiamare di “linguaggi comparati”: estesi, approfonditi, stimolanti. Ma *troppo* estesi e approfonditi (in una parola: difficili) per lo studente che inizia il suo cammino con un solo (o al massimo due) linguaggi di programmazione e che deve ancora approfondire i concetti di base.

Questo testo intende colmare questa lacuna. Gli esperti vedranno che l'indice delle materie ripercorre in larga misura i temi classici. Ma questi stessi temi sono trattati in modo elementare, cercando di assumere come prerequisiti solo il minimo indispensabile e sforzandosi di non fare un catalogo delle opzioni possibili nei diversi linguaggi di programmazione esistenti. Il lettore di riferimento è quello che conosce (bene) un linguaggio (per esempio Pascal, C, C++, o Java); meglio se ha avuto anche una qualche esposizione ad un altro linguaggio o ad un altro paradigma. Si sono evitati riferimenti consistenti a linguaggi ormai desueti e gli esempi di codice sono solo raramente espressi in uno specifico linguaggio di programmazione: il testo usa con libertà una sorta di pseudolinguaggio (ispirato nella sua sintassi concreta a C e Java), cercando di descrivere così gli aspetti più rilevanti che uniscono i diversi linguaggi.

Di tanto in tanto un “riquadro” a capo pagina presenta un approfondimento, o il richiamo di una nozione di base, o qualche dettaglio specifico dei linguaggi più comuni (C, C++, Java 5.0; ML e LISP per i linguaggi funzionali; PROLOG per i linguaggi logici). Il materiale dei riquadri può quasi sempre essere tranquillamente omesso in una prima lettura.

Tutti i capitoli presentano una breve serie di esercizi, intesi come banco di prova per la comprensione del materiale. Non vi sono esercizi davvero difficili o che richiedano più di una decina di minuti per esser risolti.

Il Capitolo 3 (Fondamenti) affronta temi che solitamente non sono trattati in un testo di linguaggi di programmazione. È tuttavia naturale, discutendo di semantica statica e confrontando tra loro linguaggi, chiedersi quali siano i limiti dell'analisi statica dei programmi e se quello che si può fare in un linguaggio si possa fare anche in un altro. Invece che rimandare ad altri testi, vista la rilevanza sia culturale che pragmatica di queste questioni, abbiamo deciso di rispondere direttamente a queste domande. In modo informale, ma rigoroso, nel contesto di poche pagine viene presentata l'indecidibilità del problema della fermata e l'equivalenza dei linguaggi di programmazione quanto alle funzioni calcolabili. Questo permette di esporre lo studente, che non sempre ha nel suo curriculum un corso completo sui “fondamenti”, ai risultati principali relativi alle limitazioni dei procedimenti di calcolo, che riteniamo fondamentali per la sua formazione.

Insieme ai principi, il testo introduce anche ai tre principali *paradigmi di programmazione*: quello orientato agli oggetti (un tema ormai obbligato della formazione dell'informatico), quello funzionale, quello logico. La necessità di realizzare un testo introduttivo e, insieme, di mantenerne l'estensione in un numero

di pagine ragionevole, spiega l'esclusione di temi importanti, quali la concorrenza e i linguaggi di scripting, che costituiscono competenze importanti ma che difficilmente possono essere affrontati correttamente assieme agli aspetti di base.

Uso del testo Il testo è in primo luogo un manuale universitario, anche se la quasi totale assenza di prerequisiti matematico-formali lo rende adatto anche allo studio personale del professionista che voglia approfondire la propria conoscenza dei meccanismi che stanno dietro ai linguaggi che utilizza. La scelta dei temi e lo stile di presentazione sono stati ampiamente influenzati dall'esperienza di insegnamento di questi contenuti presso il corso di laurea in Informatica della Facoltà di Scienze Matematiche, Fisiche e Naturali dell'*Alma Mater Studiorum* – Università di Bologna.

Nella nostra esperienza, un corso di linguaggi di programmazione di sei crediti posto al secondo anno della laurea triennale può coprire gli aspetti fondamentali (diciamo i 4/5) dei primi dieci capitoli e forse accennare brevemente ad uno dei restanti paradigmi. Al crescere della maturità degli studenti, aumenta ovviamente la quantità di materiale che può essere presentato. In una laurea magistrale il materiale potrebbe anche essere completato dalla trattazione della compilazione.

Ringraziamenti La nostra gratitudine a Giorgio Levi va ben al di là del fatto che ha avuto la bontà di scrivere la prefazione. Entrambi dobbiamo a lui le nostre prime conoscenze dei meccanismi che sottostanno ai linguaggi di programmazione: il suo insegnamento ritorna in questo libro in modo tutt'altro che marginale.

Ugo Dal Lago ha composto le figure usando METAPOST. Cinzia Di Giusto, Wilmer Ricciotti, Francesco Spegni e Paolo Tacchella hanno letto e commentato con attenzione le bozze di alcuni capitoli.