Conclusioni

Al giorno d'oggi, l'evoluzione dei linguaggi di programmazione può essere suddivisa in cinque fasi fondamentali, riguardanti rispettivamente:

- **linguaggi di prima generazione**, basati sul codice-macchina ed espressi tramite il sistema binario);
- **linguaggi di seconda generazione**, simbolici, e di tipo *assembly*, in cui le istruzioni sono rappresentate da nomi che richiamano vocaboli inglese e i dati possono essere identificati da lettere assegnate a campi di memoria centrale. Tali linguaggi sono orientati alla macchina e per il loro corretto utilizzo è necessario conoscere a fondo la struttura dell'elaboratore su cui si programma. Appartengono a tale categoria i linguaggi trattati in questa tesi;
- **linguaggi di terza generazione**, procedurali, orientati all'uomo e non alla macchina, generalmente aventi parole-chiave in lingua inglese e nei quali il programmatore deve specificare "cosa" ottenere in output e "come" ottenerlo. L'apprendimento di ciascuno di tali linguaggi è permesso anche a coloro che non conoscono il funzionamento dei componenti hardware dell'elaboratore su cui si lavora. Esempi di tali linguaggi sono *COBOL*, *FORTRAN*, *Pascal*, *C* e *BASIC*;
- **linguaggi di quarta generazione**, non procedurali, basati anch'essi sulla lingua inglese, dotati di dizionari di dati integrati o relativi a *database* relazionali dinamici (linguaggio *SQL*) e nei quali il programmatore deve specificare "cosa" ottenere in output ma lascia al software il compito di stabilire "come" ottenerlo. Tali linguaggi sono caratterizzati da una somiglianza ancora più spinta con il

linguaggio naturale, da una minore rigidità formale e dall'uso di un'interfaccia utente-elaboratore più trasparente e di alto livello. A questa categoria appartengono anche i linguaggi specifici associati a particolari prodotti software (come gestori di *database*, fogli elettronici e *word processor*) e i linguaggi di Intelligenza Artificiale, che utilizzano nuove particolari strutture;

- **linguaggi di quinta generazione**, quali quelli *object-oriented* e quelli in grado di fornire una pura interfaccia grafica per la programmazione.

Questo lavoro di tesi ha permesso di mettere in evidenza il momento del passaggio fra i linguaggi di prima generazione e quelli di seconda, identificabile con l'apporto – avvenuto nell'ottobre del 1946 e descritto nel paragrafo 2.4 – di sostanziali modifiche e migliorie alla versione iniziale dell'ENIAC, intervento che ha dotato tale macchina di un set basilare di istruzioni che permettevano all'operatore di abbandonare la mera programmazione in linguaggio binario.
Un successivo importante incremento della complessità e della portata dei set di istruzioni di cui erano dotate le macchine del periodo preso in esame si ebbe con la possibilità di specificare, oltre all'istruzione da eseguire, anche una (nel caso dell'EDSAC) o più (nel caso dell'EDVAC) locazioni di memoria nelle quali fossero contenuti i dati oggetto dell'operazione.
Fu però la comparsa dell'IBM 650 a portare alla nascita di un vero e proprio linguaggio di tipo *assembly*, caratterizzato da un numero di istruzioni molto più ampio rispetto a quante fossero a disposizione dei calcolatori immediatamente precedenti, ma soprattutto accompagnato da una nuova metodologia di programmazione in grado di garantire ampia libertà di manovra all'operatore, il quale poté – fra l'altro – definire parti di programma da riutilizzare o iniziare a preoccuparsi dell'ottimizzazione della memoria e dell'uso delle risorse di calcolo.
Se fino a questo momento i set di istruzioni disponibili erano prefissati e non passibili di modifiche, grazie al Bendix G-15 fu possibile personalizzare (per quanto in misura non eccessiva) i comandi previsti dalla macchina e adattarli a particolari esigenze di calcolo o a specifiche caratteristiche dei dati oggetto della computazione, quali per esempio lo spazio occupato nella memoria dai valori in input, che talvolta solevano essere contenuti in *double-words* e non in *single-words*.

Poco dopo la diffusione sul mercato del Bendix G-15 iniziò il processo di passaggio dai linguaggi di tipo *assembly* (detti anche *shortcodes*) a quelli di terza generazione, capaci di astrarsi dal livello-base della programmazione non binaria. Fra i primi linguaggi di terza generazione apparsi si segnalano:

- il *FORTRAN* (*FORmula TRANslator*), creato nel 1957 da John Backus dell'IBM, considerato il primo linguaggio di programmazione di alto livello e finalizzato all'automatizzazione dei calcoli matematici e scientifici;
- l'*ALGOL* (*ALGOrithmic Language*), creato nel 1958 e ricordato per avere introdotto i concetti di ciclo, di istruzione IF e di *subroutines* ricorsive;
- il *LISP* (*LISt Processor*), creato nel 1958 per lo studio delle equazioni di ricorsione e utilizzato nei progetti di Intelligenza Artificiale;
- il *COBOL* (*COmmon Business Oriented Language*), creato nel 1960, utilizzato soprattutto per sviluppare programmi atti alla risoluzione di problemi aziendali relativi a fatturazione, contabilità e stipendi e in grado di garantire una notevole facilità d'uso.