Storia dell'informatica - Prima parte

Luca Barra

Anno accademico 2023/2024

Indice

Capitolo 1

Storia delle architetture e dei sistemi di calcolo

Capitolo 2

Storia dei linguaggi di programmazione

2.1 Introduzione

L'idea di "linguaggio di programmazione" emerge per far fare al compilatore determinati compiti. Ogni computers, per quanto sofisticato comprende solo il concetto di bit, ma per gli esseri umani è difficile esprimersi in quei termini. Per questo motivo si è pensato di creare un linguaggio che fosse più vicino alla nostra comprensione. Con il tempo si sono succedute molte idee e c'è stata una selezione naturale dei linguaggi. Alla fine i linguaggi veramente importanti sono poco più di una decina.

Note:-

Ai giorni nostri si da per scontata l'idea di linguaggio, ma neggli anni '40, Von Neumann affermava di non vederne l'utilità.

All'inizio si riteneva che nessuno avrebbe deciso di scrivere un programma in un linguaggio perchè sarebbe stato troppo lento rispetto allo scrivere un programma in assembler, ma oggi quest'idea è superata per via dei sempre più efficienti compilatori.

Ripercorrendo la storia dei sistemi di calcolo si ha una prima idea con l'analytical engine di Babbage, con Ada Lovelace che scrive il primo programma per questa macchina. Successivamente con ENIAC si ha il primo computer elettronico (che poteva essere riconfigurato), ma il primo linguaggio di programmazione è il Plankalkul di Konrad Zuse, che però non è mai stato implementato. Nel Mark I Di Aiken le istruzioni erano codificate su nastro perforato. Tra il '43 e il '45 Goldstine e Von Neumann sviluppano il concetto di "flow chart" in cui si ha "=" interpretato come assegnamento. Nel '48 il Manchester SSEM usa a 32 switch per stabilire il valore di un bit. Nell'EDVAC, i bit che componevano il programma (scritto in linguaggio macchina) e i dati, tutti rappresentati in binario, erano inseriti uno a uno nelle Mercury Delay Lines.

Definizione 2.1.1: Initial order

Nel EDSAC nasce per la prima volta l'idea "initial order": il codice di una determinata istruzione era associato a una lettera assegnata in modo mnemonico (per esempio "S" significava subtract). Un istruzione macchina era costituita da un tre caratteri di 5 bit ciascuno.

Note:-

Nell'initial order il programma era codificato su un nastro perforato. Era una sorta di antenato del linguaggio assembler.

In sostanza:

- Alla fine degli anni '40 c'erano poche decine di programmatori e meno di 20 computers;
- Non esistevano corsi/tutorial di programmazione;
- Erano stati scritti pochi programmi.

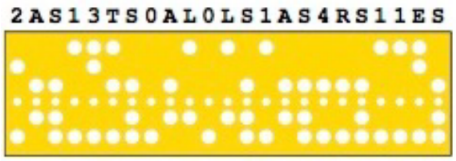

Figure 2.1: Initial order

Definizione 2.1.2: Short code

Lo short code viene implementato nel '49 da Maurice Wilkes. Si tratta di un linguaggio che permette di scrivere programmi in modo più semplice. Veniva assegnato uno specifico numero a 6 bit per indicare variabili e simboli di un'equazione. Si scrisse anche un programma per il calcolo delle equazioni da calcolare (una sorta di primitivo interprete).

Tra il 1948 e il 1950 Haskell Curry sviluppa la programmazione strutturata. Tuttavia Curry non considerava la fase di analisi sintattica

2.2 Gli anni '50

Definizione 2.2.1: For

Il costrutto for viene introdotto nel '51 da Rutishaur. La sua idea permetteva di generare codice rilocabile.

Nel 1950 l'italiano Corrado Böhm concepisce un linguaggio di alto livello e un metodo di traduzione in linguaggio macchina. Nel suo lavoro:

- Vengono introdotti "if then else" e "goto";
- Gli statement di assegnamento;
- Le subroutine;
- Un compilatore scritto nello stesso linguaggio dei programmi che deve tradurre.

Il compilatore di Böhm genera codice proporzionalmente al numero di passi da eseguire. Inoltre il linguaggio di Böhm è universale (tuttavia è solo su carta).

2.2.1 AUTOCODE

Definizione 2.2.2: AUTOCODE

Il primo compilatore degno di questo nome viene svippato nel 1953 da Alick Glennie che noto che si poteva usare lo stesso computer per tradurre un programma e far girare il programma sullo stesso computer. AUTOCODE era complicato e il suo compilatore era composto da 750 istruzioni.

AUTOCODE:

- Rendeva più veloce e semplice la programmazione;
- Costituiva una perdita di efficienza del 10% rispetto al linguaggio macchina.

Ma il lavoro di Glennie non ebbe il successo sperato poichè in quegli anni il focus non era tanto sullo scrivere un programma ma sul fatto che i calcolatori si rompessero continuamente.

Negli anni '50 si sviluppa l'idea di pseudo-codice e si parlava di automatic coding invece che di compilatori.

2.2.2 Gli anni dal '54 al '56

In queglu anni si stava tenendo il SIMATIC (Symposium on the Mechanization of Thought Processes) in cui si discuteva delle proprie scoperte.

- Nel 1954 si sviluppa il primo assembler moderno, il SOAP;
- Tra il 1954 e il 1956 alla Boeing Airplane Company di Seattle viene sviluppato il sistema BACAIC, in cui espressioni algebriche vengono tradotte in subroutine di linguaggio macchina;
- Enton Elsworth lavora ad un sistema per la traduzione di equazioni algebriche in linguaggio macchina (dell'IBM 701), e chiama il suo sistema Kompiler;
- Viene messo a punto ADES, il primo linguaggio di programmazione imperativo;
- Nel 1956 , per IBM 650, viene sviluppato il compilatore IT.

Definizione 2.2.3: IT

IT funzionava in due fasi:

1. Veniva generato codice assembler intermedio;
2. Da quel codice veniva generato codice macchina

IT permetteva di scrivere in un linguaggio semplice con un'implementazione efficiente.

2.2.3 Il FORTRAN

Definizione 2.2.4: FORTRAN

Nel 1957 il FORTRAN fa la sua comparsa. All'inizio del 1954 John Backus, Harlan Herrick e Irving Ziller iniziano a lavorare su un linguaggio di programmazione. In quegli anni i programmi venivano scritti o in assembler o in linguaggio macchina, per cui non si pensava che il FORTRAN potesse avere successo.

Specifiche:

- Avrebbe dovuto essere facile scrivere programmi in FORTRAN e sarebbe dovuto essere efficiente;
- Avrebbe dovuto essere facile da imparare;
- Non doveva essere svincolato dall'hardware (in quanto linguaggio di IBM).

Il manuale del FORTRAN aveva una grafica professionale, ma era pieno di errori e incompleto. Tuttavia il FORTRAN influenzo la maggior parte dei linguaggi successivi e fino agli anni '70 fu utilizzato come standard per applicazioni scientifiche.

2.2.4 LISP

Definizione 2.2.5: LISP

Nel 1958 fa il suo debutto il LISP. Fu il primo dei linguaggi funzionali. Nasce per la manipolazione di espessioni simboliche con l'uso del concetto di "ricorsione". Inoltre il LISP usava sia liste che alberi. Offriva un garbage collector e un sistema di tipi dinamico. Permetteva la meta-programmazione.

- Viene usata una notazione polacca (prefissa);
- Alcuni operatori potevano venire implementati direttamente in linguaggio macchina.

Note:-

Alcune parti di LISP erano codificate in FORTRAN

In LISP tutto veniva rappresentato da liste concatenate. La lista concatenata a destra è la rappresentazione interna dei dati.

2.2.5 COBOL

Definizione 2.2.6: COBOL

Il COBOL (COmmon Business Oriented Language) fu concepito per applicazioni di tipo amministrativo e commerciale. Il COBOL aveva una sintassi "english-like", ciò lo rendeva facilmente comprensibile. Esempio:

ADD A TO B GIVING C

Il COBOL venne sviluppato dal CODASYL (Conference on Data Systems Languages) e il DoD (Department of Defense) USA obbligo le compagnie produttrici di computer a fornire il COBOL nelle loro installazioni.

Corollario 2.2.1 La sintassi del COBOL

La sintassi del COBOL fu il risultato di un processo decisionale influenzato da FLOW-MATIC (di Grace Hopper) e dall'IBM COMTRAN (di Bob Bemer)[a].

[a] Molto marginalmente.

Gli obiettivi:

- doveva essere portatile;
- doveva essere efficiente;
- doveva essere facilmente comprensibile e utilizzabile.

Ma il COBOL non fu recepito bene dalla comunità informatica: non si consideravano i programmatori COBOL dei veri programmatori. Inoltre il COBOL non era adatto per applicazioni scientifiche. Ai giorni nostri il COBOL è quasi del tutto scomparso.

2.3 Gli anni '60

2.3.1 ALGOL 60

Definizione 2.3.1: L'ALGOL 60

L'ALGOL 60 (ALGOrithmic Language 1960) fu frutto di una collaborazione tra l'ACM (Association for Computing Machinery) e l'IFIP (International Federation for Information Processing). L'ALGOL 60 era molto legato all'hardware sottostante.

Note:-

Tony Hoare[a] noto che l'ALGOL 60 era molto più avanti rispetto ai linguaggi dell'epoca.

[a] Inventore del Quicksort e del puntatore NULL

ALGOL 60 introduce:

- **call by value** (per i parametri);
- **call by name** (per le variabili);
- **call by reference** (per i parametri, passaggio di un puntatore);

L'ALGOL è il primo linguaggio a richiedere esplicitamente il tipo delle variabili, Inoltre viene introdotta la distinzione tra assegnamento e uguaglianza. Oltre a qeusto si introducono le prime primitive di controllo strutturate: if then else, while loop e for. Si introducono anche i blocchi di istruzioni:

BEGIN ... END

L'ALGOL per via del pesante utilizzo di GOTO consentiva programmazione non strutturata. Ciò rende il codice meno leggibile.

Definizione 2.3.2: Spaghetti code

Spaghetti code indica codice non strutturato e "ingarbugliato", difficilmente leggibile.

2.3.2 La Backus-Naur Form (BNF)

Definizione 2.3.3: Backus-Naur Form (BNF)

La BNF è una notazione per grammatiche context free. Viene usata per descrivere la sintassi dei linguaggi di programmazione e, essendo un metà linguaggio, può essere usata per descrivere se stessa.

Note:-

Tutto ciò viene spiegato nel corso "Linguaggi formali e traduttori"

Definizione 2.3.4: Panini Backus Form

La Panini Backus Form è una versione ridotta della BNF.

Corollario 2.3.1 | parsificatori

I parsificatori sono programmi che leggono un testo e lo analizzano per determinare la sua struttura grammaticale. ALGOL porto alla creazione di un parsificatore (LL). Nel 1965, Donald Knuth sviluppa un altro parsificatore (LR), ma solo alla fine degli anni '60 verranno messi a punto dei parsificatori efficienti.

2.3.3 BASIC

Definizione 2.3.5: Basic (Beginner's All-purpose Symbolic Instruction Code)

Il BASIC nasce nel 1964 da un progetto di John Kemeny e Thomas Kurtz. Il BASIC era un linguaggio di programmazione semplice e facile da imparare. Il suo obiettivo era quello di consentire a studenti, non di informatica, di scrivere programmi.

Note:-

Il BASIC inizia a diffondersi negli anni '70 con la diffusione dei "micro-computers". In alcuni casi il BASIC diventava l'ambiente di lavoro del PC (un po' come la shell).

Il BASIC era sufficientemente ad alto livello da poter essere utilizzato da chiunque ed ebbe diffusione come linguaggio didattico. Ma anche il BASIC fu guardato con sufficienza da molti informatici: Dijstra sosteneva che fosse inutile insegnare a programmare a gente che era stata esposta al COBOL o al BASIC.

Definizione 2.3.6: Visual BASIC

Il visual BASIC fu introdotto da Microsoft nel 1991 sebbene fu utilizzato molto poc a causa della diffusione di linguaggi più potenti.

Altre varianti furono:

- Altair BASIC;
- Game BASIC (poi ridenominato Integer BASIC);
- Applesoft BASIC.

2.3.4 La questione del GOTO

Teorema 2.3.1 Bohm-Jacopini

Il teorema ha diverse formulazioni equivalenti, ma sostanzialmente asserisce che qualsiasi funzione computabile può essere calcolata da un programma costituito esclusivamente da una combinazione di:

- Sequenze di istruzioni eseguite una dopo l'altra;
- Istruzioni di selezione (del tipo if then else);
- Istruzioni di iterazione (del tipo while do).

Note:-

In sostanza il teorema dimostrava che il GOTO poteva essere messo da parte.

Nel 1968, Dijstra critica l'uso del GOTO in una lettera alle Communication of the ACM. Successivamente, nel 1974, Knuth mostra che in alcuni casi il GOTO era la scelta migliore. Ne manuale di Brian Kernigham e Dennis Ritchie osservano che il GOTO era necessario per gestire alcuni casi di errore.

2.3.5 Il Simula

Definizione 2.3.7: Simula I

Il Simula I nasce nel 1966 da un progetto di Kristen Nygaard e Ole-Johan Dahl. Il Simula I è il primo linguaggio ad oggetti. Il suo scopo principale era quello di gestire simulazioni.

Note:-

Dahl e Nygaard si ispirarono al concetto di record class introdotto da Hoare nell'ALGOL 60.

Definizione 2.3.8: Simula 67

Il Simula 67 è la seconda versione del Simula. Il Simula 67 è il primo linguaggio ad oggetti a essere usato in ambito commerciale. Il Simula 67 introduce il concetto di classe e di sottoclasse. Le istanzee di una classe sono dette oggetti e l'operazione new permette di creare un oggetto. In Simula 67 erano anche presenti i puntatori (chiamati ref) ai record di attivazione (oggetti).

Note:-

Per questi motivi si introduce un meccanismo di garbage collection.

Il Simula 67 introduce anche la data abstraction e l'ereditarietà. Uno degli utilizzi del simula era la modellazione di sistemi concorrenti.

2.4 Gli anni '70

2.4.1 Pascal

Definizione 2.4.1: Pascal

Il Pascal nasce nel 1970 da un progetto di Niklaus Wirth. Può essere considerato un successore dell'ALGOL. Vengono introdotti i puntatori e il tipo CHAR. Il Pascal incoraggia la costruzione di procedure chiare, usa strutture dati dinamiche, call by value e call b reference. Un'altra caratteristica era il suo sistema di tipi molto ricco, ma molto rigido[a].

[a] Con type checking in fase di compilazione

Obiettivi:

- Semplicità;
- Chiarezza;
- Efficienza;
- Scrittura di programmi complessi.

Il compilatore del Pascal non generava direttamente codice macchina, ma generava un codice intermedio: il P-Code. Il P-Code era un linguaggio assembly portabile. Il P-Code veniva interpretato da una macchina virtuale. Verso la fine degli anni '70 il Pascal era ormai in uso e insegnato nelle università.

L'Educational Testing Service (ETS), rese il Pascal il linguaggio standard per i test AP (Advanced Placement). Nel 1983 Hejlsberf sviluppa il Turbo Pascal, un compilatore per PC. Il Turbo Pascal era un compilatore molto efficiente e permetteva di scrivere programmi in Pascal in modo molto veloce (diventò lo standard dei PC).

Apple utilizzo il Pascal come linguaggio di riferimetno per il suo sistema operativo così come Microsoft. Tuttavia il Pascal declino in favore di C e di UNIX.

2.4.2 Il Prolog

Definizione 2.4.2: Il Prolog

Il Prolog (PROgrammation en LOGique) nasce nel 1972 da un progetto di Alain Colmerauer e Philippe Roussel. Il Prolog è un linguaggio dichiarativo basato sulla logica del primo ordine. Il Prolog è un dimostratore automatico di teoremi.

Note:-

I dimostratori automatici di teoremi sono utilizzati nel corso di "Metodi formali per l'informatica" e nella parte da +3 CFU del corso di "Linguaggi e paradigmi".

Capitolo 3

Storia dei sistemi operativi