Introduzione ai linguaggi di programmazione

Come ormai ben sappiamo, il software permette a un computer di svolgere tutti quei compiti così diversi tra loro che ne fanno una macchina universale, capace di scrivere un testo o risolvere un'equazione, giocare una partita di scacchi o controllare altre macchine. Ogni specifico software è costituito da un **programma**, ovvero da una successione di istruzioni che vengono eseguite una dopo l'altra dal computer.
Poiché sono proprio queste istruzioni che permettono ai componenti hardware della macchina di svolgere i compiti loro assegnati, un computer sprovvisto del software è del tutto inutilizzabile e si riduce a un semplice ammasso di plastica e metallo.

Algoritmi e programmi

Prima di predisporre un software occorre svolgere l'analisi del problema che l'elaboratore deve risolvere e determinare così un algoritmo adeguato. Il termine "algoritmo" deriva dal soprannome (al-Khuwarizmi) del matematico arabo al-Muhammad ibn Musa, vissuto nel nono secolo d.C., che in un suo celebre trattato (al-Giabr, da cui la parola "algebra") descrisse le procedure per risolvere alcuni problemi della matematica del tempo.
Nel corso del medioevo il nome al-Khuwarizmi (distorto in “algoritmo”), passò quindi a significare un qualsiasi procedimento che indicasse le operazioni da eseguire per ottenere, a partire dai dati disponibili, i risultati voluti.
Esempi di algoritmi sono quelli per calcolare una radice quadrata o il massimo comun denominatore, o il minimo comune multiplo di due o più numeri.

Naturalmente un algoritmo scritto in italiano, ma anche in inglese, latino o in un qualsiasi altro linguaggio naturale, come in informatica vengono chiamate le lingue usate dall'uomo, non può essere eseguito direttamente dal computer.
Per questo occorre prima trascriverlo in un linguaggio formale che l'elaboratore sia in grado di interpretare ed eseguire correttamente. Un programma non è altro che un algoritmo trascritto in un linguaggio formale, che per questa ragione è anche detto linguaggio di programmazione

Le moltissime lingue che l’uomo ha elaborato, negli oltre 100.000 anni della sua esistenza, sono tutte caratterizzate da un alto grado di ambiguità. Prendiamo ad esempio la frase “ho visto un uomo con un cannocchiale”:essa può significare, a seconda dei casi, “ho visto un uomo che teneva in mano un cannocchiale” oppure “guardando attraverso un cannocchiale ho visto un uomo". Entrambe le interpretazioni sono perfettamente coerenti e quando parliamo con qualcuno siamo in grado di capire, dal contesto in cui la frase è pronunciata, quale delle due sia

corretta. Oppure pensiamo all'ambiguità dell'espressione "la giovane mente"… Purtroppo gli elaboratori, per quanto potenti, non sono ancora in grado di sciogliere l'ambiguità e individuare l'interpretazione esatta. Qualsiasi istruzione venga loro fornita deve quindi essere scritta in un linguaggio governato da regole grammaticali precise e che non contenga alcuna forma d'ambiguità. Questi linguaggi sono appunto i linguaggi di programmazione.

In realtà le sole istruzioni che un computer è in grado di interpretare ed eseguire sono quelle espresse in linguaggio macchina, un linguaggio in cui i programmi sono rappresentati da una lunga sequenza di cifre binarie che codificano i comandi e i dati su cui lavora la CPU.

Come è facile immaginare, un programma scritto direttamente in linguaggio macchina è pressoché incomprensibile anche al suo stesso autore; se i computer dovessero essere programmati direttamente in questo linguaggio il loro utilizzo sarebbe estremamente problematico.

Un primo, parziale passo verso la semplificazione della programmazione è stato dato con l'uso dei linguaggi assembler (assemblatori), in cui le singole istruzioni binarie sono rappresentate con un codice più facilmente comprensibile per l'uomo.

A partire dagli anni Cinquanta, con la realizzazione dei primi computer commerciali, sono stati poi sviluppati particolari linguaggi di programmazione il cui impiego risulta molto più semplice dei linguaggi assembler.

Questi linguaggi, sebbene sottoposti come tutti i linguaggi formali a ferree regole di composizione, utilizzano simboli e parole tipiche delle lingue naturali (soprattutto dell'inglese), di facile interpretazione.

Per questo motivo sono infinitamente più simili al linguaggio naturale non solo delle successioni apparentemente senza senso di 0 e 1 del linguaggio macchina, ma anche dei programmi assembler; per questa ragione vengono detti linguaggi evoluti o di alto livello. Appositi programmi detti compilatori, provvedono quindi a tradurre le istruzioni scritte nei linguaggi evoluti nel loro equivalente in linguaggio macchina.

La compilazione, eseguita da un software chiamato appunto compilatore, traduce il programma sorgente, ovvero il programma scritto utilizzando l'editor, in programma oggetto, cioè in programma scritto nel linguaggio macchina ed eseguibile dall'elaboratore.

Spesso il programma fa riferimento ad una parte del software che non viene elaborata direttamente dal programmatore, ma che è reperibile nelle librerie di sistema.

Questo software viene collegato al programma nella fase di link da un insieme di programmi indicati con il termine collettivo di linker.

Durante queste fasi, oltre a tradurre il programma sorgente nel corrispondente programma oggetto, i compilatori provvedono anche a trovare, e segnalare, gli errori di carattere sintattico presenti nel programma.
Se nella compilazione vengono evidenziati alcuni errori, il programma oggetto non può essere prodotto e dobbiamo tornare alla fase di editing per correggerli.

Durante l'esecuzione il programma oggetto viene caricato in memoria centrale ed eseguito. In questa fase potranno essere rilevati gli errori non sintattici, come ad esempio la mancanza di sufficiente memoria per l'esecuzione del programma oppure la divisione di un numero per zero; errori che non possono essere individuati dalla compilazione. In questi casi l'esecuzione viene interrotta.
Sempre durante l'esecuzione, il programmatore ha la possibilità di verificare se il programma opera correttamente oppure se sono stati commessi errori logici, per cui i risultati ottenuti non sono quelli desiderati.
Al termine dell'esecuzione è in ogni caso possibile tornare alla fase di editing, correggere gli errori ed eventualmente modificare o ampliare il programma per migliorarne l'efficienza.

Un software che gestisce le tre fasi della programmazione, collegando direttamente i vari ambienti, viene chiamato ambiente di sviluppo integrato.

I Linguaggi di Programmazione

Il primo a redigerne uno fu Alan Turing, che voleva rendere più facile l'uso del Manchester Mark 1.
Il linguaggio da lui creato conteneva cinquanta istruzioni, che erano automaticamente trascritte in codice binario dallo stesso computer.
Una delle prime grandi figure della programmazione fu però una donna, Grace Murray Hopper. Ella sviluppò con l'UNIVAC1 ciò che l'omonima compagnia chiamava la "programmazione automatica": un programma interno che trasformava le istruzioni dell'utente in istruzioni per la macchina, tradotte in codice binario.
Il primo vero linguaggio di programmazione fu il FORTRAN.
Il problema, con i linguaggi di programmazione, di cui il FORTRAN fu il capostipite, era che il computer non era stato concepito per leggerne ed eseguirne direttamente le istruzioni. Di conseguenza, bisognava redigere un programma intermedio, chiamato compilatore, che traduceva tutto il programma: ad esempio, ciò che era scritto in FORTRAN era tradotto nel linguaggio codificato della macchina. In pratica, il FORTRAN poteva funzionare con qualunque computer, ma,

per ogni modello, bisognava elaborare un compilatore adatto, che riconoscesse sia il codice di una determinata macchina che le istruzione del linguaggio di programmazione impiegato.
Il programma sorgente viene elaborato dai traduttori, che hanno la funzione di leggere e tradurre le informazioni comunicandole poi al computer .

Tra i programmi traduttori ci sono:

- i **compilatori** : traducono contemporaneamente tutte le sequenze informative nel linguaggio macchina specifico; le informazioni tradotte non vengono però eseguite ed eventuali errori non possono essere visualizzati e corretti se non controllando l'intero programma alla fine della compilazione. Il compilatore ha il pregio di avere un'elevata velocità di esecuzione.
- gli **interpreti** : fanno la traduzione riga per riga senza memorizzare ciò che hanno prodotto , si limitano ad eseguirlo nella stessa modalità(riga per riga). Gli interpreti sono più lenti dei compilatori però segnalano subito eventuali errori presenti nelle sequenze. Non e' possibile portare da un processore ad un altro ciò che e stato tradotto, in quanto il programma eseguibile dipende dall'interprete.

Il primo compilatore scritto per il FORTRAN, da usarsi con un IBM 704, venne elaborato nell'aprile del 1957 e comportava 25000 linee di programma. Grazie ad altri compilatori, il FORTRAN poté essere ben presto usato su altre macchine e non solo su quelle IBM. Questo linguaggio era particolarmente adatto ai calcoli scientifici ed era di facile impiego in confronto al modo in cui fino a quel momento erano stati usati i computer.

Indipendentemente dalle ulteriori versioni, che hanno reso il FORTRAN più efficiente, sono stati elaborati altri linguaggi di programmazione, in funzione di specifici settori.
Il COBOL viene riservato, come indica il nome, alle applicazione nel settore terziario. L'ALGOL, un linguaggio molto tecnico elaborato nel 1960, viene usato poco ma è ampiamente studiato come modello. Il PL/1 (Programming Language number 1) venne completato nel 1964 e soddisfa allo stesso tempo le necessità dei settori serviti dal FORTRAN e dal COBOL: di conseguenza si tratta di un linguaggio universale. Il PL/1 ha ricevuto alcune critiche perché risulta difficile verificare durante l'uso se alcuni programmi eseguono correttamente gli ordini impartiti.

Il LISP (1956) e, in seguito, il PROLOG, insieme al PASCAL, hanno portato qualche progresso in materia di programmazione. La grande novità sarà il BASIC, che si rivolgerà essenzialmente ai non iniziati. Elaborato per scrivere brevi programmi, una istruzione dopo l'altra, grazie al suo sistema di linee accuratamente numerate, il BASIC sembrerà perdere la sua efficacia nell'approccio con insiemi più complessi. Il C, per la sua grande efficienza, è stato largamente utilizzato per sviluppare interi sistemi operativi, come UNIX o Linux. Con la certificazione da parte dell'ANSI

(American National Standards Institute) nel 1988, il C si può definire il primo linguaggio standardizzato con la possibilità di "portare" programmi da un sistema ad un altro. Vi sono poi dei programmi realizzati per soddisfare esigenze specifiche, come per esempio quelli per risolvere problemi matematici, di algebra, di statistica, di calcolo ed altri.
I linguaggi di programmazione sono andati evolvendosi nel tempo e sono, grazie all'impiego di sempre nuove tecniche, migliorati nel corso degli anni. Si possono suddividere i vari linguaggi di programmazione che si sono avvicendati negli anni in cinque categorie, chiamate anche generazioni. Questo termine non e' giustificabile solo come suddivisione cronologica, ma indica anche la maggior importanza del nuovo tipo di linguaggio rispetto al tipo di applicazione e al problema da risolvere. Vediamo ora in particolare le caratteristiche delle cinque generazioni:

I Linguaggi della prima generazione
I linguaggi della prima generazione sono i cosiddetti "linguaggi macchina", perché si basano solamente su sequenze di 0 ed 1 (bits), usati direttamente dal processore per interpretare i comandi. Ogni processore quindi utilizza un linguaggio macchina differente. Una caratteristica negativa di questi linguaggi è che l'utilizzo di istruzioni basate su sequenze 0,1 comporta una notevole difficoltà sia nella stesura del programma che nella sua successiva interpretazione. Bisogna però evidenziare che ogni linguaggio di programmazione, anche il più evoluto, altro non ha che il compito di tradurre vari tipi di istruzioni in linguaggio macchina, unica forma comprensibile dal computer.

I Linguaggi della seconda generazione
I linguaggi della seconda generazione, hanno come caratteristica principale quella di aver sostituito alle interminabili sequenze 0,1, alcuni termini di più facile comprensione, come ad esempio l'alfabeto ed i numeri in base decimale. Il linguaggio che rappresenta questa seconda generazione è l'**assembler**. A questo livello però la sostituzione è ancora molto elementare, in quanto i comandi sono ancora costruiti allo stesso modo del linguaggio macchina. Anche questo tipo di linguaggio come quelli della prima generazione risulta però di difficile uso e comprensione per l'uomo.

I Linguaggi della terza generazione
I linguaggi di questa generazione sono anche chiamati linguaggi di alto livello o linguaggi avanzati. Essi danno un certo grado di "indipendenza" dal microprocessore e sono dotati di una serie di comandi più complessi rispetto alle generazioni precedenti, che permettono di risolvere in maniera molto più semplice vari tipi di problemi. Viene di molto semplificata anche la struttura di programmazione, permettendo all'operatore di suddividere il programma in blocchi ciascuno con funzioni specifiche richiamabili quando necessario. A differenza dell'assembler, ora i comandi non sono più scritti secondo uno schema che segue il

linguaggio macchina, dunque il traduttore si trasforma da assemblatore a **compilatore o interprete**.

I Linguaggi della quarta generazione

La descrittività del linguaggio usato è la caratteristica fondamentale dei linguaggi della quarta generazione. Sono la conseguenza logica delle prime tre generazioni, le quali passo dopo passo hanno cercato di avvicinarsi il più possibile al linguaggio umano. Questi linguaggi basano la loro potenza e versatilità su comandi predisposti, i quali svolgono funzioni anche molto complesse prima svolte da più comandi di linguaggio avanzato.
Spesso si identificano i linguaggi legati all'intelligenza artificiale come i linguaggi della quarta (e quinta) generazione.

I Linguaggi della quinta generazione

Questi linguaggi vengono prevalentemente impiegati nell' ambito dell'intelligenza artificiale, che ha come scopo quello di imitare artificialmente i comportamenti intelligenti attribuibili solamente all'uomo. La grande categoria dei linguaggi della quinta generazione e' suddivisibile in tre distinti gruppi:

1. linguaggi logici: si basano sulle teorie della logica matematica e il più rappresentativo e' il PROLOG.
2. linguaggi basati su oggetti: si basano sulla definizione di oggetti anziché dei tradizionali algoritmi. Devono il loro successo e la loro diffusione alla nuova concezione di utilizzazione di "icone" per quanto riguarda l'uso dell'elaboratore.
3. linguaggi funzionali: ricostruiscono in pratica tutte le possibili funzioni utilizzando come punto di partenza le tre funzioni base, e cioè il concatenamento, l'iterazione e la ricorsione.

Linguaggi di programmazione ad alto livello.

Alcuni dei linguaggi ad alto livello più antichi e diffusi sono: il Fortran, il Cobol, il Pascal, il Basic.
In commercio esistono numerosi linguaggi per poter comunicare con la macchina, alcuni di questi sono più vicini al linguaggio fatto di cifre binarie, che è usato direttamente dalla macchina, i cosiddetti "Linguaggio macchina".
Questi linguaggi vengono chiamati "Assemblatori" o "Assembler"; essi hanno istruzioni molto semplici, ciascuna delle quali traduce in forma simbolica, con sigle ed espressioni di facile comprensione , una istruzione in linguaggio macchina.

Gli assembler sono programmi che un normale utilizzatore del computer non è in grado di operare, infatti questi sono pur sempre linguaggi per specialisti. Per gli utenti meno specializzati sono stati creati dei linguaggi di programmazione chiamati "Linguaggi ad alto livello", in questi linguaggi ogni istruzione è espressa in

forma o in simboli molto più intuibile, e la struttura è molto più vicina al linguaggio umano che no a quello della macchina, in alcuni di questi linguaggi si può utilizzare sia il tipo programmazione in forma algoritmica, e sia quella ad oggetti .
Per esempio nel Pascal c'è l'istruzione "write" (cioè se si traduce significa scrivi), il cui effetto è quello di far scrivere al calcolatore su un monitor i dati che nel programma seguono la funzione stessa.(Esempio delle differenze tra Assembler e linguaggi ad alto livello "Pascal")

In principio era il bit...
In principio era il bit. Eh sì, per poter "insegnare" al computer come svolgere qualche semplice operazione aritmetica bisognava essere in grado di programmare in linguaggio macchina (0 e 1 per intenderci).

Tra gli anni '40 e gli anni '50, la programmazione degli elaboratori elettronici era destinata solamente a una ristrettissima cerchia di esperti. E quello che può essere considerato il progenitore di tutti i computer è sicuramente l'ENIAC (Electronic Numerical Integrator And Calculator), un mastodontico calcolatore pesante 30 tonnellate, al cui progetto partecipò il brillante matematico J. von Neumann.

Noi ora, "viziati" dalle semplici tecniche di programmazione di VisualBasic sicuramente sorridiamo di pensando all'ENIAC, per quel tempo era una vera e propria innovazione. Infatti fu il primo elaboratore programmabile interamente a circuiti elettronici e senza parti meccaniche in movimento, ma questa è un'altra storia...

Fortran (FORmula TRANslation)
Elaborato per l'IBM 701, nel 1957, ad opera del gruppo di John Backus (un dipendente dell'IBM), fu uno dei primi linguaggi di programmazione. Lo scopo principale del FORTRAN era quello di automatizzare calcoli matematici e scientifici. Questo linguaggio ebbe molto successo e ha subito nel corso degli anni numerose modifiche e aggiornamenti per adeguarsi progressivamente alle esigenze delle nuove macchine. Sulla sua scia vennero progettati moltissimi altri linguaggi di alto livello.

Algol
Questo linguaggio (ALGOrithmic Language) interessò molti studiosi, tra i quali lo stesso Backus, che, insieme al matematico Naur, mise a punto in quel periodo un sistema per rappresentare le regole dei linguaggi (Backus-Naur Form). Non si ebbe però lo stesso successo ottenuto con il FORTRAN.

Cobol (COmmon Business Oriented Language)
Nato nel 1960 per iniziativa del governo americano, dei grandi utenti e delle principali case produttrici di elaboratori, con l'obiettivo di sviluppare un linguaggio che potesse essere impiegato nel maggior numero possibile di macchine , serviva

soprattutto per sviluppare programmi gestionali, cioè atti alla risoluzione di problemi aziendali (fatturazione, contabilità, stipendi etc.).
Uno dei motivi della sua enorme diffusione è stato la sua discreta facilità, dovuta anche al fatto che le istruzioni somigliano molto a frasi inglesi. Ad esempio, la frase "aggiungi il valore di importo al valore totale" si tradurrebbe in COBOL così: "add import to totale". Grazie all'appoggio del governo USA, il COBOL divenne ben presto il linguaggio più diffuso per le applicazioni di tipo commerciale e gestionale.
E' tuttora di grande importanza per le numerose librerie di vecchi programmi che, magari ad alcuni decenni dalla loro compilazione, continuano a svolgere il loro lavoro installati in computer di banche, ministeri e multinazionali.

Basic (Beginners All purpose Symbolic Instruction Code).

(codice generale di istruzioni simboliche per i principianti) É nato grazie al progetto di Kurtz e Kemeny in un college americano per avvicinare gli studenti alla programmazione nel modo più semplice possibile nel 1964 ed era quindi rivolto ai principianti.
Si tratta quindi di un linguaggio relativamente facile da imparare e di uso piuttosto comune; per queste ragioni il BASIC si diffuse rapidamente fra gli studenti diventando, alla fine degli anni Settanta, il primo linguaggio di alto livello a essere applicato ai personal computer.
A tutt'oggi è tra i migliori linguaggi utilizzati nelle scuole a scopo didattico e, dato il favore incontrato, ne sono state realizzate molte versioni per apportare miglioramenti. Sul Macintosh abbiamo il Chipmunk BASIC che è rimasto piuttosto fedele alla sintassi dei suoi "antenati".
Inizialmente usate soprattutto dai programmatori dilettanti, alcune versioni potenziate del BASIC sono oggi largamente impiegate anche dai programmatori professionisti. Si tratta in genere di versioni che includono gli strumenti della programmazione visuale (Visual BASIC), in cui l'interazione fra l'utente e la macchina avviene principalmente attraverso elementi grafici.

Pascal

Nel 1971 Niklus Wirth, un professore del politecnico di Zurigo, ideò e realizzo il linguaggio Pascal (così chiamato in onore del matematico e filosofo francese Blaise Pascal) affinché facilitasse l'applicazione delle regole e della tecniche di programmazione.
Il suo scopo era ottenere un linguaggio adatto per l'insegnamento della scrittura di programmi e centrò benissimo il suo obiettivo, tanto che il Pascal è ancora oggi molto usato nelle scuole.
Lo scopo del Pascal era quello di superare le limitazioni dei precedenti linguaggi, obbligando gli utenti a seguire delle procedure corrette e standardizzate di programmazione.
I punti forti del Pascal sono quindi un'impeccabile definizione dei dati usati e la facilità con cui è possibile controllare la sequenza delle istruzioni. Per queste ragioni il Pascal rimane un linguaggio particolarmente adatto ai fini didattici, mentre la sua

impostazione può risultare un po' limitativa se applicata a programmi di tipo commerciale.

C

Nel 1972 Dennis Ritchie presso i laboratori Bell, realizzò la prima versione del linguaggio C sviluppando un precedente linguaggio noto come B, ed è stato impiegato per scrivere il sistema operativo Unix, in modo che i suoi utenti potessero modificarlo e migliorarlo autonomamente. Il C si distingueva dai suoi predecessori per il fatto di implementare una vasta gamma di tipi di dati come carattere, interi, numeri in virgola mobile, strutture etc. Da allora il linguaggio non ha subito profonde trasformazioni: la sua sintassi é stata estesa, soprattutto in conseguenza della programmazione orientata agli oggetti (C++), ma nella sostanza il linguaggio e' rimasto quello delle origini.
Fra le principali caratteristiche del C figura la portabilità, ovvero la possibilità che hanno i programmi scritti in questo linguaggio di essere installati su macchine diverse con minimi cambiamenti. Il C è soprattutto un linguaggio estremamente versatile, con cui è possibile scrivere programmi impiegati in ogni possibile campo di applicazione; proprio grazie a questa versatilità, il C è diventato un linguaggio estremamente diffuso soprattutto fra i programmatori professionisti. Questi pregi del C sono anche accompagnati da una certa difficoltà connessa al suo apprendimento.
Il C e' un linguaggio ad alto livello che possiede un insieme ristretto di costrutti di controllo e di parole chiave, e un ricco insieme di operatori, ed è un linguaggio apparentemente povero: non possiede istruzioni di entrata/uscita nè istruzioni per operazioni matematiche. È stato talvolta definito come "il linguaggio di più basso livello tra i linguaggi ad alto livello", infatti nasce per lo sviluppo di sistemi operativi, quindi per software di basso livello, ma riesce a mantenere semplicità d'uso. Il trucco usato dai suoi sviluppatori per mantenere compatto il linguaggio, e nel contempo estenderne le funzionalità, sta nell'affidare le funzioni più complesse a un'insieme di librerie esterne: esattamente come il MacOS, al quale, per aggiungere funzionalità aggiungiamo estensioni e librerie.
Alcuni anni dopo l'introduzione del C venne sviluppata una successiva e più potente versione nota con il nome di C++ (da Bjarne Stroustrup nel 1983). Con il C++ è possibile utilizzare le tecniche della programmazione orientata agli oggetti (OOP) un nuovo e potente metodo di programmazione che integra dati e procedure in un unico modulo software: l'oggetto.
Il concetto di oggetto (o più precisamente di classe) è semplice, dividere l'interfaccia dal contenuto, ottenendo in questo modo tanti "moduli" interagibili tra loro attraverso le interfacce, permettendo così al programmatore di cambiare il contenuto di una classe (se sono stati trovati errori o solo per introdurre delle ottimizzazioni) senza per questo doversi preoccupare di controllare eventuale altro codice che richiami la classe. Come si può intuire tale linguaggio ha completamente stravolto il modo di programmare precedente ad esso che, sostanzialmente, si

riduceva ad una programmazione procedurale che lasciava poco spazio al riutilizzo del codice; basti pensare

Java

Java, la cui creazione risale a metà degli anni '90, è stata un'invenzione rivoluzionaria e indispensabile nel mondo della programmazione e non solo per i browser. La SUN, qualche anno fa presentò il Java come un linguaggio di programmazione Object Oriented semplice e familiare a chi conosce il C++, indipendente dall'architettura, e sicuro per l'uso in rete. In effetti, uno stesso programma funziona su Mac, PC , Unix e SGI, senza richiedere modifiche o ricompilazioni. Però, per chi non conosce il C++, Java risulta difficile da imparare come qualsiasi altro linguaggio. Essendo un linguaggio multipiattaforma, il compilatore non crea degli applicativi eseguibili (APPL nel Mac o .exe in Windows) dal computer, ma dei file .class che devono essere interpretati dalla JVM (Java Virtual Machine, un microprocessore virtuale) implementata sui vari sistemi. Ciò ha sempre reso l'esecuzione di un'applicazione Java molto più lenta dei programmi normali, e solo a partire dalla versione 3 di Netscape Navigator ed Explorer, grazie al compilatore Just In Time, l'esecuzione ha iniziato ad avere tempi accettabili. Dal punto di vista economico-aziendale, il Java, si è rivelato vincente, abbattendo i costi e il tempo per la conversione di programmi C/C++ per piattaforme differenti. Distinguiamo gli applet, ovvero i programmi che si eseguono all'interno delle pagine HTML, e quelli eseguibili solo con l'interprete Java, che sono vere applicazioni.

Altri linguaggi di programmazione da menzionare sono il **LISP** (1959), l'**ADA** (1970), lo **SMALLTALK** (1970) e il **LOGO**. Negli ultimi cinque anni, inoltre, si sono fatti strada linguaggi di scripting come l'**ASP** e il **PHP**; i "vecchi" linguaggi, come il Basic, il Pascal o il C++, sono diventati "VISUALI" e hanno aperto la strada ad una programmazione più intuitiva e veloce.