Storia dei linguaggi di programmazione

"A computer will do what you tell it to do,
but that may be much different from what you had in mind"
Joseph Weizembaum

Giovanni Riccardi

Il primo linguaggio

Charles Babbage

Ada Lovelace, discepola e collaboratrice di Babbage, definì il primo linguaggio di programmazione (~1837).

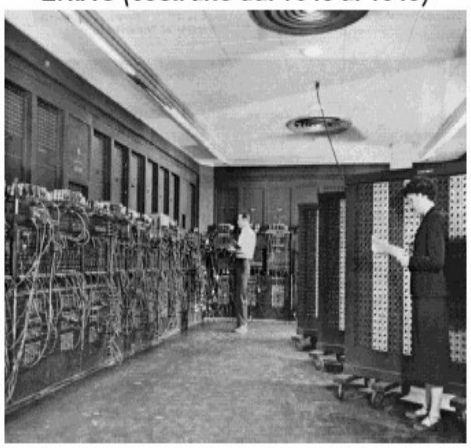
Ada Lovelace

- Era un linguaggio di tipo assemblativo
- Introdusse il concetto di ciclo ripetuto e il concetto di variabile indice.
- La realizzazione della macchina analitica non fu mai portata a termine.

"Quando la macchina analitica verrà realizzata, necessariamente guiderà lo sviluppo futuro della scienza."
Charles Babbage

Dopo più di cento anni (’43)

ENIAC (costruito dal 1943 al 1945)



La programmazione dell’ENIAC era definito da circuiti elettrici.

- La macchina non aveva un programma registrato: sei donne erano impegnate a muovere commutatori e connettere cavi.

Valido anche come pittore, a lato il suo autoritratto. - Konrad Zuse

Gli anni della guerra

Konrad Zuse,
ingegnere tedesco,
sviluppò il Plankalkül (1943-5).

- Sviluppò il linguaggio mentre se ne stava nascosto sulle Alpi della Baviera in attesa della fine della Seconda Guerra Mondiale.
- Usò il suo linguaggio come opponente nel gioco degli scacchi sul suo computer Z3.
- Il linguaggio era già in grado di gestire sia tabelle che strutture di dati.
- Il Plankalkül rimase seppellito in qualche archivio in Germania per molto tempo.

I linguaggi macchina (’50)

Una sequenza di bit.

- LOAD 8 potrà essere rappresentata in una macchina reale come 00110100, dove 0011 è la rappresentazione interna del codice operativo LOAD
- I linguaggi di programmazione coincidevano con l'insieme delle istruzioni eseguibili dall'hardware.
- Enorme sforzo richiesto per codificare algoritmi semplici.

I problemi dei linguaggi macchina

- Sono specifici della macchina.
  - Ogni CPU ha il proprio linguaggio macchina.
  - Occorre conoscere l’architettura della macchina per scrivere programmi.
  - I programmi non sono portabili.
- I codici sono illeggibili all’uomo.
- I programmatori si specializzano nel cercare efficienza su una macchina specifica, anziché concentrarsi sul problema.

La prima evoluzione

Codifica di tipo simbolico, anziché binaria, dei programmi:

"linguaggi assembler"

- Per la prima volta con la nascita degli assembler fu applicato il principio che è meglio risparmiare il tempo dell'uomo anche a costo di sprecare tempo-macchina (una parte del tempo è dedicata alla traduzione di programmi, non alla loro esecuzione diretta).

I linguaggi assembly

- In assembly ogni istruzione è identificata da una sigla piuttosto che da un numero e le variabili sono rappresentate da nomi piuttosto che da numeri.
- I programmi scritti in assembly necessitano di un apposito programma assemblatore per tradurre le istruzioni tipiche del linguaggio in istruzioni macchina.
- Oggi si utilizza l'assembly solo se esistono vincoli stringenti sui tempi di esecuzione.

I problemi dell'assembly

- Sono comunque legati all'architettura della macchina.
- I linguaggi assembly non sono sufficienti a gestire l'enorme complessità dei programmi moderni.
- TOP_DOWN o BOTTOM_UP ?
  - Il modo naturale di procedere è pensare prima alla struttura generale e poi curare i dettagli ...
  - Ma questo è impossibile con l'Assembly che è fatto SOLO da dettagli...

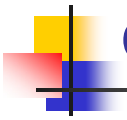
Il passo successivo

Rendere la codifica degli algoritmi il più possibile "vicina" al problema da risolvere anziché all'architettura della macchina.

- Avere programmi scritti in formalismi comprensibili a un largo numero di sviluppatori.
- Indipendenza da architetture specifiche.

I Linguaggi ad alto livello

- Tra gli anni ’50 e ’60 si passo ai linguaggi ad alto livello.
- Essi richiedono un compilatore o un interprete che sia in grado di tradurre le istruzioni del linguaggio di alto livello in istruzioni macchina di basso livello eseguibili dal calcolatore
- Un compilatore è un programma simile ad un assemblatore, ma più complesso, infatti ...
  - esiste una corrispondenza biunivoca fra istruzioni in assembler ed istruzioni macchina
  - ogni singola istruzione di un linguaggio di alto livello corrisponde a molte istruzioni in linguaggio macchina.



Gli anni ’50

- FORTRAN:
  - Primo linguaggio ad utilizzare un compilatore
- COBOL
  - Uno dei più importanti linguaggi procedurali
- LISP
  - Linguaggio per l’elaborazione simbolica.

John Backus

Il FORTRAN

- Il FORmula TRANslator è stato sviluppato da un gruppo di programmatori della IBM guidati da John Backus e pubblicato per la prima volta nel 1957.
- Era stato progettato per facilitare la traduzione in codice di formule matematiche.

IBM 704

Il FORTRAN

- Data la sua semplicità di scrittura i programmatori riuscivano ad essere fino a 500 volte più veloci in FORTRAN che in altri linguaggi.
- E' stato il primo linguaggio "problem oriented" anziché "machine oriented".

'La Nonnina del COBOL ″- Grace Murray Hopper

Il COBOL

- Sviluppato nel 1959 da un gruppo di professionisti riuniti alla Conference on Data Systems Languages (CODASYL).
- Da allora ha avuto molte modifiche e miglioramenti. Nel tentativo di superare le numerose incompatibilità tra le varie versioni, l'American National Standards Institute (ANSI) annunciò uno standard nel 1968, che produsse l'ASN-COBOL.
- Il linguaggio continua ad evolversi ancora oggi, che è disponibile in una versione object-oriented inclusa nel COBOL 2002

John McCarthy

Il LISP

- Nel 1958 John McCarthy fu incaricato di creare una lista di specifiche per creare l'elaborazione simbolica.
- Il List Processing Language fu integrato come estensione nel FORTRAN stesso.

Guy L. Steele, Jr. now working on Java

Il LISP

Un programma LISP può modificare se stesso crescendo

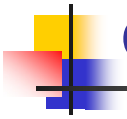
- Sia i dati sia i programmi sono delle liste
- Per questo motivo un programma in LISP può trattare altri programmi al suo interno.
- E' utilizzato nello studio della intelligenza artificiale.
- Contiene il concetto di Garbage Collector (cf JAVA)

LISP: "Lots of Irritating Superfluous Parentheses"

Una delle prime Lisp Machine al MIT Museum

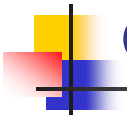
Lisp Machine

- Nel 1970 venne costruito un computer speciale (chiamato Lisp Machine) in grado di elaborare programmi LISP.
- Da notare come la tastiera contiene dei simboli propri del linguaggio.



Gli anni ’60

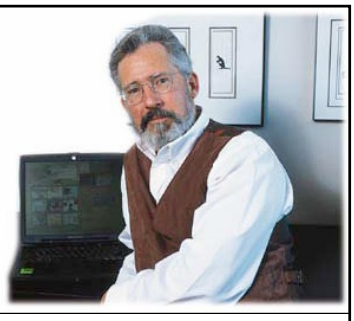
- Algol-60:
  - Primo linguaggio con una sintassi formalmente definita (BNF)
  - Primo linguaggio basato sui principi della programmazione strutturata
- Simula-67
  - Introduce il concetto di classe e oggetto



Gli anni ’70

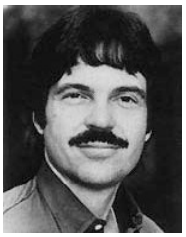
- Pascal:
  - Sviluppato per l’insegnamento
  - Ha avuto un grande successo anche nel mondo dell’industria
- PROLOG:
  - Introduce la programmazione logica
- Smalltalk:
  - Prime interfacce grafiche
  - Object Oriented
- C:
  - Concepito inizialmente come una sorta di assembler strutturato.
  - E’ diventato il linguaggio più affermato nella programmazione di sistema.

Dan Ingalls



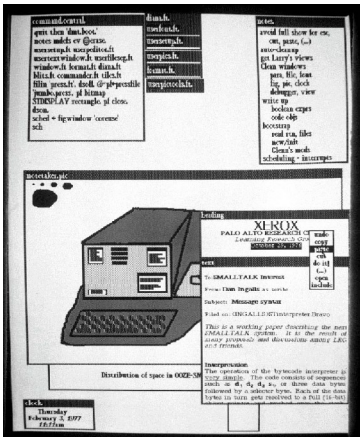
Lo Smalltalk

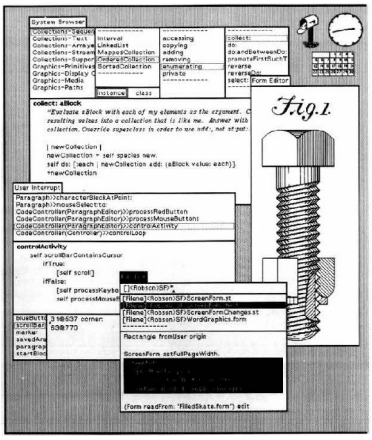
- Realizzato (1972) da Alan Kay e Dan Ingalls a Xerox PARC (Palo Alto Center of Research)
- Fortemente Object Oriented
- Introdusse la "reflection"
- Con esso sono state realizzate le prime interfacce "windows"



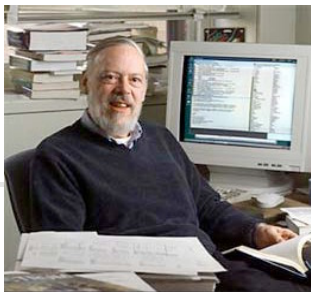
Alan Kay

Smalltalk: esempi di interfacce





Dannis Ritchie



Il linguaggio C

- Il linguaggio C fu sviluppato al Bell Laboratories nel ′72 da Dennis Ritchie.
- Molti dei suoi principi e idee erano presi dal precedente linguaggio B e il B aveva ereditato certe caratteristiche da BCPL e CPL.
- La potenza e flessibilità del C apparve subito evidente e per questo il sistema operativo di Unix, scritto in assembly, venne riscritto immediatamente in C.

"The C Programming Language″ Brian Kernighan

Il linguaggio C

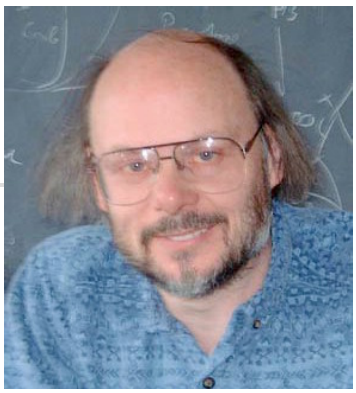
- Fino alla fine degli anni '70, il C invase molti college e università per il suo stretto legame con Unix e la disponibilità di un compilatore.
- La definizione formale si ha nel 1978 a cura di B. W. Kernighan e D. M. Ritchie.
- L'ANSI (American Standards Institute) nel 1983 formò un comitato per stabilire una definizione standard del C (ANSI-C).
- Oggi il C è diffusissimo, insieme ad una ricca raccolta di librerie di funzioni.

Gli anni ’80

- Ada:
  - Omaggio alla prima programmatrice
  - ADA doveva rappresentare il punto di maturazione perfetta di tutti i principi di costruzione del software e dei relativi meccanismi linguistici elaborati negli anni precedenti.
- C++:
  - Uno dei migliori linguaggi Object Oriented

Il C++

Bjarne Stroustrup



- E' un'estensione del C, sviluppato da Bjarne Stroustrup all'inizio degli anni '80, presso i Bell Laboratories.
- Fornisce le capacità di gestione della programmazione di tipo object-oriented.
- E’ utilizzato per lo sviluppo di importanti SW: prodotti Adobe, prodotti Microsoft, molti prodotti Apple, etc.

Gli anni ’90

- Java:
  - Il linguaggio più utilizzato al mondo

Il Java

James Gosling

- Il linguaggio Java è derivato da un linguaggio chiamato OAK, che fu sviluppato nei primi anni '90 alla Sun Microsystem come linguaggio piattaforma-indipendente predisposto per applicazioni di intrattenimento come console per video game e VCR per comunicazioni.
- OAK fu impiegato per la TV via cavo, per ordinare i programmi da vedere.
- Mentre quel tipo di spettacolo on-demand tramontava, il World Wide Web, invece, riscontrava sempre più interesse. A quel punto i tecnici sviluppatori di OAK ci si buttarono a capofitto, trasformando il programma OAK nel nuovo Java.

Curiosità sul Java

- J. Gosling e A. Van Hoof si trovavano spesso ad un caffè presso il quale discutevano del linguaggio stesso. E così il linguaggio prese il nome da tale abitudine (Java è una qualità di caffè dell'omonima isola dell'Indonesia).
- Il magic number che identifica un file **.class** è **0xCAFEBABE** (probabilmente riferendosi alla cameriera che li serviva).

Microsoft vs. Java

La Microsoft propose alla Sun un accordo per apportare modifiche al linguaggio, la Sun rifiuta, allora Microsoft dichiara guerra alla Sun sviluppando il linguaggio C# (siamo nel 2000), sviluppato sulla base di JAVA (ovvero java con le modifiche che Microsoft aveva proposto alla Sun che rifiutò).

Oggi ...

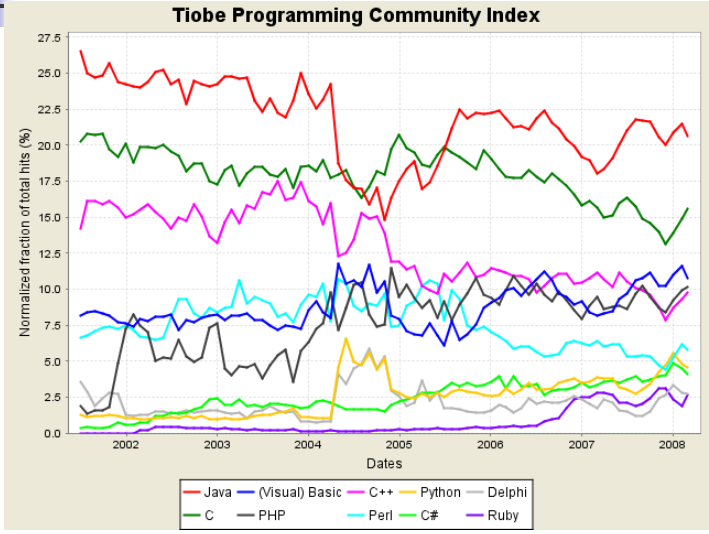
Sul sito dell'HOPL

(History of Programming Languages)

"hopl.murdoch.edu.au/home.prx"

Sono contenute informazioni su **8.512** linguaggi di programmazione.

La top 10

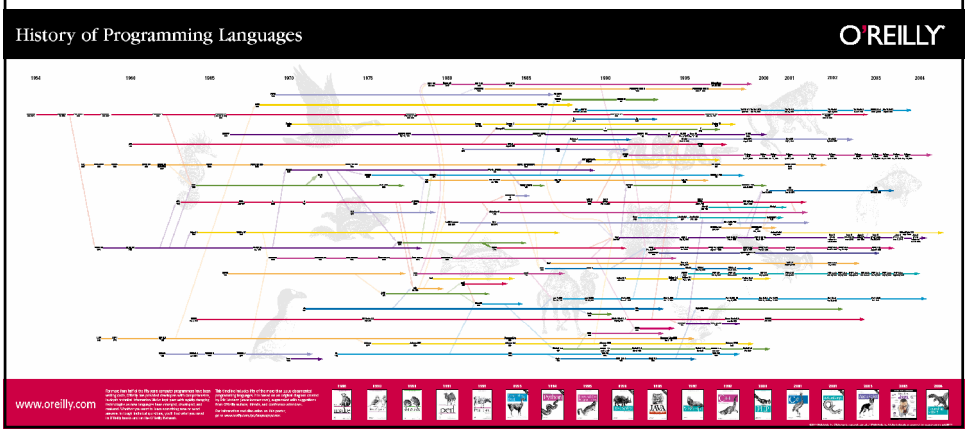


Category Ratings

	March 2008	Delta March 2007
Object-Oriented Languages	54.9%	+3.0%
Procedural Languages	42.8%	-1.4%
Functional Languages	1.6%	-0.6%
Logical Languages	0.7%	-1.0%

Il diagramma di Éric Lévénez

- Sono rappresentati solo 50 linguaggi

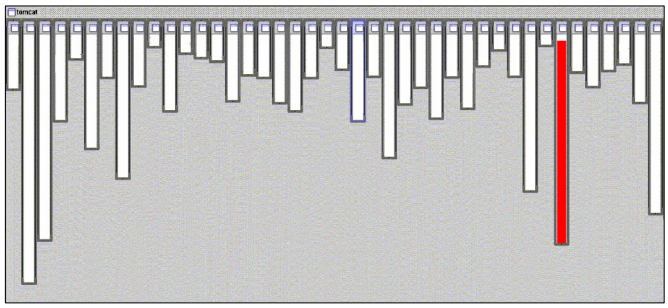


Programmazione Aspect Oriented

- La nascita dell'Aspect Oriented Programming (AOP) si deve a XEROX PARC (Palo Alto Research Center) negli anni 90.
- Concetto "trasversale" alla programmazione ad oggetti (anche se si integra con essa) in quanto ci aiuta ad affrontare in maniera "elegante" problematiche che riguardano l'applicazione nel suo insieme.

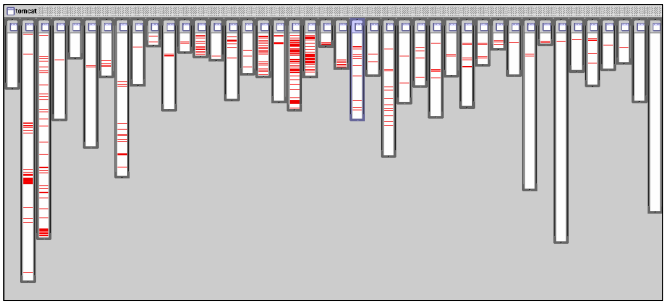
Esempio di Concern modulare

Osservando il codice di org.apache.tomcat notiamo che funzionalità come l'XML parsing (sono colorate in rosso le parti di codice interessate) vengono ben modellate con la OOP.



Esempio di concern di sistema

Osserviamo invece che il codice che gestisce il logging viene distribuito in quasi tutte le classi. Questo è un classico esempio di crosscutting concern.



Programmazione Aspect Oriented

- L'AOP si occupa di trasformare in "moduli" i concern di sistema (crosscut concern) similmente a come il java trasforma in "moduli" (classi) i concern di modulo.

Gregor Kiczales

Un linguaggio AOP: AspectJ

- E' una estensione AOP di Java, sviluppato da Gregor Kiczales ed il suo gruppo alla Xerox PARC.
- AspectJ aggiunge a java il concetto di join point.

I join point (pointcut in AspectJ)

- Identificano dei punti nell'esecuzione di un programma.
- Esempi pointcut:

  call(void Point.setX(int))

  Identifica tutti i punti in cui viene chiamato il metodo setX(int) della classe Point.

  call(void Figure.make*(..))

  Identifica tutti i punti in cui viene chiamato un metodo il cui n ome inizia per make della classe Figure.

Gregor Kiczales

AspectJ

- Esempio di aspect

```
aspect Logging {
    OutputStream logStream = System.err;
    pointcut set():
        call(void Point.setX(int)) || call(void Point.setY(int));
    before(): set() {
        logStream.println("about to set");
    }
}
```

Bibliografia

- www.wikipedia.it
- www.windoweb.it
- http://www.research.att.com/~bs/applications.html
- Introduzione al progetto di compilatori – (Dino Mandrioli – 1988 Franco Angeli Editore)
- www.tiobe.com
- http://www.dis.uniroma1.it/~degiacomo
- Dennis M. Ritchie's homepage
  http://cm.bell-labs.com/cm/cs/who/dmr/
- Aspect Oriented programming con AspectJ
  http://www.objectway.it/articles/0308_AOPAspectJ.pdf