Unità di apprendimento 6

Il software: dal linguaggio alla applicazione

Unità di apprendimento 6
Lezione 2

Conosciamo i linguaggi di programmazione

In questa lezione impareremo:

- che cos'è un linguaggio di programmazione
- quale relazione esiste tra algoritmo e programma
- che cosa sono i codici sorgente, assemblativo ed eseguibile
- quali sono i diversi linguaggi di programmazione

Linguaggi informatici

- Quando l'esecutore di algoritmo è un elaboratore elettronico la formulazione deve essere precisa e codificata in un linguaggio comprensibile agli esecutori automatici.
- L'algoritmo deve essere trasformato in un insieme di comandi (o istruzioni elementari) che l'esecutore è in grado di eseguire in modo univoco.

Linguaggi informatici

- L'insieme delle istruzioni scritte in un linguaggio prende il nome di programma e il linguaggio relativo è il linguaggio di programmazione.
- Un programma per calcolatore, o semplicemente programma, è l'implementazione di un algoritmo.
- Viene indicato con il termine "software".

Linguaggi informatici

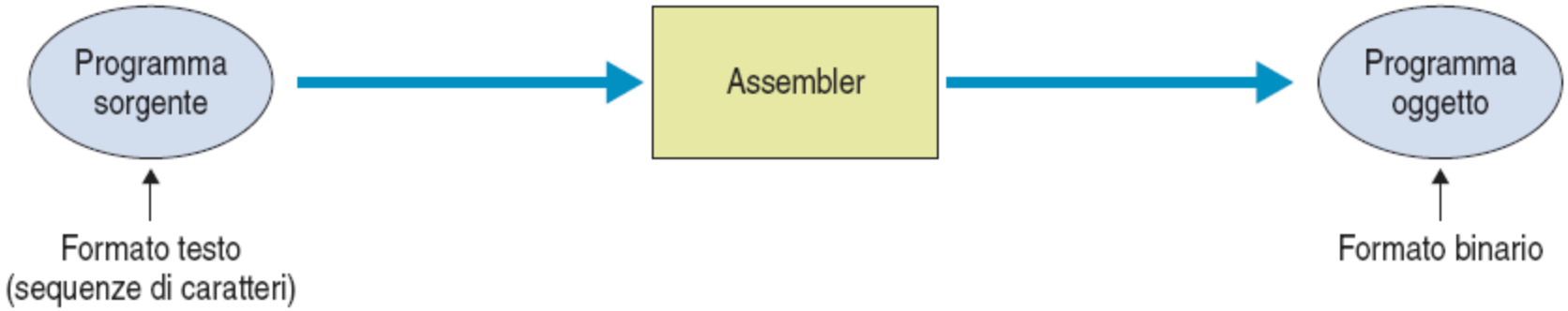
- L’algoritmo codificato in pseudocodifica è ancora molto lontano dal programma che viene eseguito dal calcolatore.
- Il microprocessore è in grado di utilizzare solo il codice binario (distinguere 0 da 1).
- Può eseguire solo un codice rappresentato con zero e uno, che si chiama codice macchina.

Linguaggi informatici

- È possibile scrivere un programma direttamente in codice macchina.
- Si utilizza un linguaggio specifica, il linguaggio assemblativo (assembly).
- Codificare programmi in questo linguaggio è molto complesso dato che mette a disposizione del programmatore poche istruzioni e di basso livello.

Linguaggi informatici

- Il programma nel suo formato originale alfanumerico (sequenza di caratteri) è chiamato programma sorgente, mentre il programma assemblato in linguaggio macchina è detto programma oggetto.



Linguaggi informatici

- L'assembly non è un linguaggio portabile, essendo strettamente legato all'architettura sulla quale opera.
- Le istruzioni macchina sono specifiche del processore (CPU).
- Ogni processore ha un suo linguaggio assembly che lo differenzia proprio perché legato al set di istruzioni del processore stesso.

Linguaggi informatici

- Esempio di codice eseguibile scritto in linguaggio macchina con a fianco il segmento di codice scritto in linguaggio assemblativo.

Indirizzo	Opcode	Data	Significato
0000	1101	000000001001	IN X
0001	1101	000000001010	IN Y
0010	0000	000000001001	LOAD X
0011	0111	000000001010	COMPARE Y
0100	1001	000000000111	JUMPGT DONE
0101	1110	000000001001	OUT X
0110	1000	000000000000	JUMP LOOP
0111	1110	000000001010	OUT Y
1000	1111	000000000000	HALT
1001	0000	000000000000	CONST 0
Codice macchina			Linguaggio assemblativo

Linguaggi ad alto livello

- E’ percorribile un’altra strada per arrivare al codice macchina:
  - scrivere l’algoritmo in un linguaggio di programmazione di alto livello;
  - mediante uno specifico programma, il compilatore, tradurre automaticamente il “programma sorgente” scritto ad alto livello in linguaggio macchina.

Linguaggi ad alto livello

- Si traduce l'algoritmo in linguaggio di programmazione ad alto livello.
- Questo è un linguaggio formale, rigoroso, composto da un insieme di regole lessicali e sintattiche molto ridotte e schematizzate in numero.
- Devono essere sufficienti per poter descrivere in modo non ambiguo le istruzioni che il calcolatore deve eseguire.

Linguaggi ad alto livello

- Esistono molteplici linguaggi di programmazione, diversi tra loro per:
  - metodologia e filosofia adottata (paradigma);
  - complessità e numero di istruzioni;
  - settore e ambito di applicazione.

LINGUAGGI A BASSO LIVELLO	Vicini alla logica della macchina e lontani da una visione umana dell'utilizzazione. Il linguaggio a più basso livello è il linguaggio macchina (cioè il binario, che è l'unico sistema di segni che il calcolatore riesce a interpretare). I programmatori che lavorano a questo livello usano l'assembler, una forma simbolica di programmazione strettamente legata al linguaggio macchina. A questo livello è necessaria una conoscenza dettagliata dell'hardware del sistema e del sistema operativo.
LINGUAGGI AD ALTO LIVELLO	Permettono una maggiore astrazione rispetto alla macchina e un uso di frammenti di frasi (in inglese) per esprimere le istruzioni (per esempio, if ... then ... else ...). Tipici linguaggi ad alto livello sono: C++, Visual Basic, Java, Objective C ecc. Per questi linguaggi non sono sempre necessarie conoscenze relative ai dettagli hardware e software del sistema su cui si opera. Talvolta, linguaggi come il C permettono una visione a basso livello del problema.
LINGUAGGI AD ALTISSIMO LIVELLO	Sono i linguaggi che non richiedono assolutamente alcuna conoscenza dei dettagli del sistema; in alcuni casi non richiedono neanche di esprimere i programmi in forma algoritmica (per esempio, un programma Prolog è composto solo da una descrizione dei domini dei dati su cui si opera, delle regole logiche di deduzione e degli obiettivi che si vogliono raggiungere; a tutto il resto provvede il linguaggio). A questa categoria appartengono i linguaggi per l'intelligenza artificiale (Lisp, Prolog), alcuni linguaggi di tipo script a oggetti (linguaggi speciali associati ad applicazioni come database, browser ecc.).
LINGUAGGI PER IL WEB	Sono i linguaggi che consentono di elaborare e rappresentare le informazioni sul Web. Si suddividono in linguaggi di markup che consentono di descrivere e rappresentare le informazioni dei siti Web (HTML, XML) e linguaggi di scripting (JavaScript, PHP, ASP e Perl) che consentono di elaborare informazioni prima di essere visualizzate dal browser.

Linguaggi ad alto livello

- Tra tutti ricordiamo solamente:
  - il linguaggio Pascal, elaborato nel 1968 dal professor Wirth del Politecnico di Zurigo, che è tutt'oggi il linguaggio più diffuso e utilizzato per affrontare lo studio della programmazione;
  - il linguaggio C, messo a punto da Dennis Ritchie per implementare i primi sistemi operativi negli anni *Settanta*, che è il linguaggio di riferimento sia per i programmatori "più esperti".

Compilatori e interpreti

- La trasformazione del programma sorgente in programma eseguibile dal processore avviene sostanzialmente secondo due alternative:
  - compilazione (con linguaggi come C, C++, FORTRAN, Pascal ecc.);
  - interpretazione (con linguaggi come Basic, Perl, JavaScript ecc.).

Compilatori e interpreti

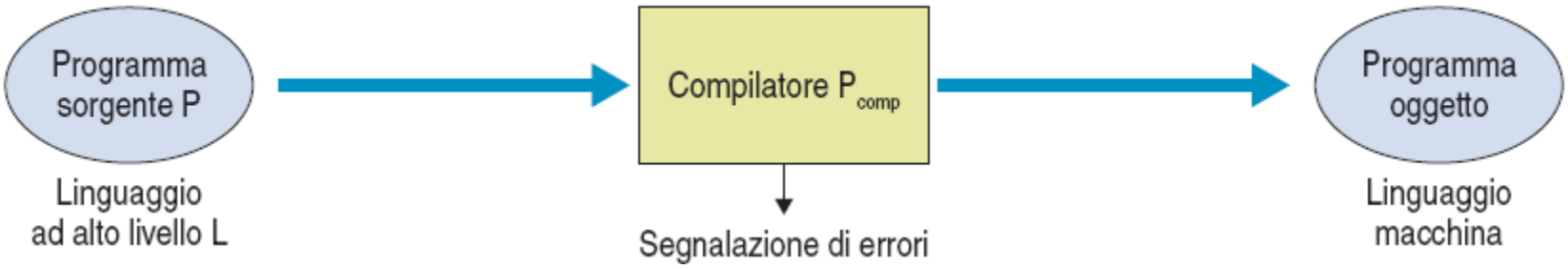
- La principale differenza tra le due modalità di traduzione è che:
  - i compilatori traducono un intero programma dal linguaggio sorgente al linguaggio macchina della macchina prescelta;
  - gli interpreti traducono e immediatamente eseguono il programma istruzione per istruzione.

Compilatori e interpreti

- nella compilazione:
  - traduzione ed esecuzione sono fatte in tempi differenti;
  - al termine della compilazione è disponibile la versione tradotta del programma in un file specifico (generalmente un file con suffisso .exe);

Compilatori e interpreti

- nella compilazione:
  - la versione tradotta è però specifica della macchina sulla quale il programma è stato compilato;
  - per eseguire il programma basta avere disponibile la versione tradotta in formato eseguibile



Compilatori e interpreti

- nell'interpretazione:
  - traduzione ed esecuzione procedono *contemporaneamente*, una di seguito all'altra;
  - al termine dell'interpretazione non vi è alcuna versione tradotta del programma originale;
  - se si vuole rieseguire il programma occorre ripetere tutto daccapo, cioè *ritradurlo*.

Compilatori e interpreti

- Linguaggi compilati
- La trasformazione dal codice sorgente al codice macchina mediante compilazione avviene attraverso un programma, appunto il compilatore.
- Legge il codice sorgente ed effettua la conversione in codice macchina mediante una sequenza di attività.

Compilatori e interpreti

- Linguaggi compilati
  - preprocessing
  - analisi lessicale (scanning)
  - analisi sintattica (parsing)
  - analisi semantica
  - generazione del codice
  - ottimizzazione del codice

Compilatori e interpreti

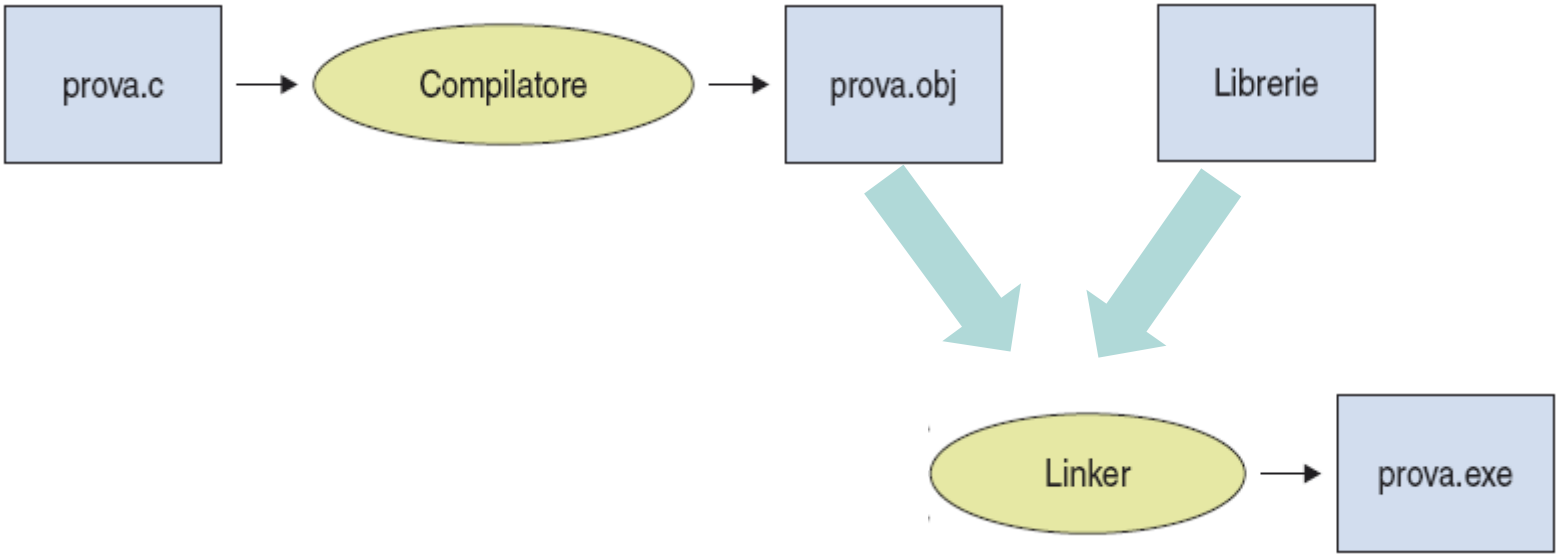
- Durante queste fasi viene rilevato dal compilatore un insieme di errori (*compile-time errors*) di tipo statico (lessicali, sintattici o semantici).
  - *fatal errors*: errori gravi che non consentono di proseguire
  - *warnings* (avvertimenti)

Compilatori e interpreti

- Come già detto in precedenza per l'assembly, il programma oggetto non è ancora pronto per essere mandato in esecuzione;
- Deve essere trasformato in .exe dal linker.

Compilatori e interpreti

- Processo di traduzione e creazione di un eseguibile



Compilatori e interpreti

- Linguaggi interpretati
- La trasformazione dal codice sorgente al codice eseguibile mediante l'interprete avviene riga per riga.
- L'interprete prende un'istruzione alla volta, l'analizza, la trasforma in codice macchina e la manda in esecuzione direttamente; comando

Compilatori e interpreti

- Viene così a crearsi un'alternanza tra la fase di traduzione e la fase di esecuzione dell'istruzione appena elaborata.
- Il vantaggio dell'interpretazione rispetto alla compilazione sta nella fase di debug in quanto vengono immediatamente individuati sia gli errori statici sia quelli di run-time.

Compilatori e interpreti

- I linguaggi interpretati sono maggiormente portabili di quelli compilati
- Richiedono un interprete specifico per ogni diversa piattaforma dove devono essere eseguiti.
- Hanno come difetto principale la lentezza
- Esempi: JavaScript, PHP, gli script della shell o a riga di comando

L a classificazione dei linguaggi in paradigmi

- Insieme di idee scientifiche collettivamente accettato per dare un senso al mondo dei fenomeni.

L a classificazione dei linguaggi in paradigmi

- I linguaggi di programmazione vengono classificati a seconda del modello computazionale che adottano.
- Al modello computazionale è stato dato il nome di paradigma, il cui significato scientifico è stato definito recentemente

L a classificazione dei linguaggi in paradigmi

- Nell'informatica un paradigma di programmazione è un modello concettuale che fornisce la "struttura" di un programma.
  - la programmazione imperativa;
  - la programmazione object-oriented (OOP);
  - la programmazione funzionale;
  - la programmazione logica

L a classificazione dei linguaggi in paradigmi

- Modello procedurale: in questo paradigma rientrano i linguaggi di tipo algoritmico;
  - imperativo: è quello classico in cui rientrano i linguaggi macchina e i linguaggi ad alto livello a partire da quelli più datati (tipo FORTRAN, COBOL, BASIC, Pascal, C) fino a quelli più recenti (JavaScript, PHP);
  - a oggetti: è un'evoluzione di quello imperativo e ha come capostipite Smalltalk ed Eiffel (linguaggi puri). Oggi abbiamo inoltre alcuni linguaggi come C++ e Java che sono "ibridi"

L a classificazione dei linguaggi in paradigmi

- Modello dichiarativo: in esso non ci si preoccupa di trovare la soluzione mediante l'ideazione dell'algoritmo, ma si procede affrontando il problema dal punto di vista descrittivo.

- Rientrano in questo paradigma
  - Modello funzionale
  - Modello logico

L a classificazione dei linguaggi in paradigmi

- Modello funzionale
  - Ha come origine il linguaggio introdotto negli anni ’30 dal matematico Alonzo Church come formalismo universale (il lambda-calcolo)
  - In esso che “tutto è un’espressione”, e come tale denota un valore ottenibile da un processo di valutazione.
  - Il nome “funzionale” deriva dal fatto che ogni simbolo di operazione che compare in un’espressione corrisponde a una funzione.
  - I principali linguaggi funzionali sono il Lisp, l’Haskell e lo Scheme.

L a classificazione dei linguaggi in paradigmi

- Modello logico
  - E’ un paradigma che si basa sulla logica dei predicati applicando le regole di inferenza del calcolo proposizionale
  - Il linguaggio “simbolo” è il Prolog (contrazione del francese PROgrammation en LOGique).
  - Oggi la programmazione logica trova molte applicazioni nell’intelligenza artificiale e nei sistemi esperti.

ABBIAMO IMPARATO CHE...

- L'insieme delle istruzioni scritte in un linguaggio prende il nome di **programma** e il linguaggio relativo è il linguaggio di programmazione.
- Il linguaggio che il processore può eseguire si chiama linguaggio macchina ed è composto da soli 0 e 1, cioè in codice binario.
- Il linguaggio assembly è una rappresentazione simbolica del linguaggio macchina ed esiste un rapporto 1:1 con le istruzioni in linguaggio macchina.
- I linguaggi di programmazione sono linguaggi simbolici che utilizzano un gruppo di parole, dette parole chiave, che possono essere combinate secondo una determinata sintassi, *per scrivere i programmi in una forma molto vicina al modo di ragionare dell'uomo.*
- La trasformazione del programma sorgente in programma eseguibile dal processore avviene con la:
  - compilazione (con linguaggi come **C, C++, FORTRAN, Pascal** ecc.);
  - interpretazione (con linguaggi come **Basic, Perl, JavaScript** ecc.).
- La compilazione consiste in una sequenza di attività, ciascuna accompagnata dalla rilevazione degli errori, qui:
  - preprocessing;
  - analisi lessicale (scanning): riconosce i lessemi (o atomi lessicali) del linguaggio sorgente;
  - analisi sintattica (parsing): organizza le frasi e segnala le sottosequenze errate;
  - analisi semantica: esegue la verifica della consistenza del "significato";
  - generazione del codice: esegue la traduzione vera e propria in linguaggio assemblativo;
  - ottimizzazione del codice: ne migliora il codice per ridurre i tempi di esecuzione.
- Il compilatore rileva un insieme di errori (compile-time errors) di tipo statico (lessicali, sintattici o semantici):
  - fatal errors: errori gravi che non consentono di proseguire la compilazione del sorgente;
  - warnings (avvertimenti): segnalazioni di errori lievi che non compromettono la traduzione.

ABBIAMO IMPARATO CHE...

- Un paradigma è un insieme di idee scientifiche collettivamente accettato per dare un senso al mondo dei fenomeni.

- Oggi i paradigmi di programmazione citati con una certa frequenza sono almeno una decina, anche se non sempre sono chiari i loro confini o la loro indipendenza. I principali sono i seguenti:
  - paradigma procedurale
    - programmazione imperativa
    - programmazione object-oriented
  - paradigma descrittivo
    - programmazione funzionale
    - programmazione logica