Informatica per l'Ingegneria

Corsi M – N

A.A. 2023/2024

Angelo Cardellicchio

12 – Linguaggi di programmazione

Cosa è un linguaggio

- **Definizione 1**: *Un linguaggio è un **insieme di parole** e di **metodi di combinazione sulle stesse** usati e compresi da una comunità di persone.*
- È una definizione poco precisa perché…
  - …non evita le ambiguità dei linguaggi naturali;
  - …non si presta a descrivere processi computazionali automatici;
  - …non aiuta a stabilire proprietà.
- **Definizione 2:** Il linguaggio è un sistema matematico che consente di rispondere a domande come:
  - quali sono le frasi lecite?
  - Si può stabilire se una frase appartiene al linguaggio?
  - Come si stabilisce il significato di una frase?
  - Quali sono gli elementi linguistici primitivi?

Sintassi e semantica

- **Sintassi:** l'insieme di regole formali per la scrittura di frasi in un linguaggio, che stabiliscono cioè la grammatica del linguaggio stesso.
- **Semantica:** l'insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio.
- **Nota:** una frase può essere sintatticamente corretta e tuttavia non avere significato!
- Le sintassi sono normalmente espresse attraverso notazioni formali, come la Backus-Naur Form o la Extended Backus – Naur Form.

Esempio: sintassi di un numero naturale

$$< naturale >::= 0| < cifra - non - nulla > \{< cifra >\}$$

- Questa notazione significa che un numero naturale può essere scritto come 0 oppure come una cifra non nulla seguita da zero o più cifre.

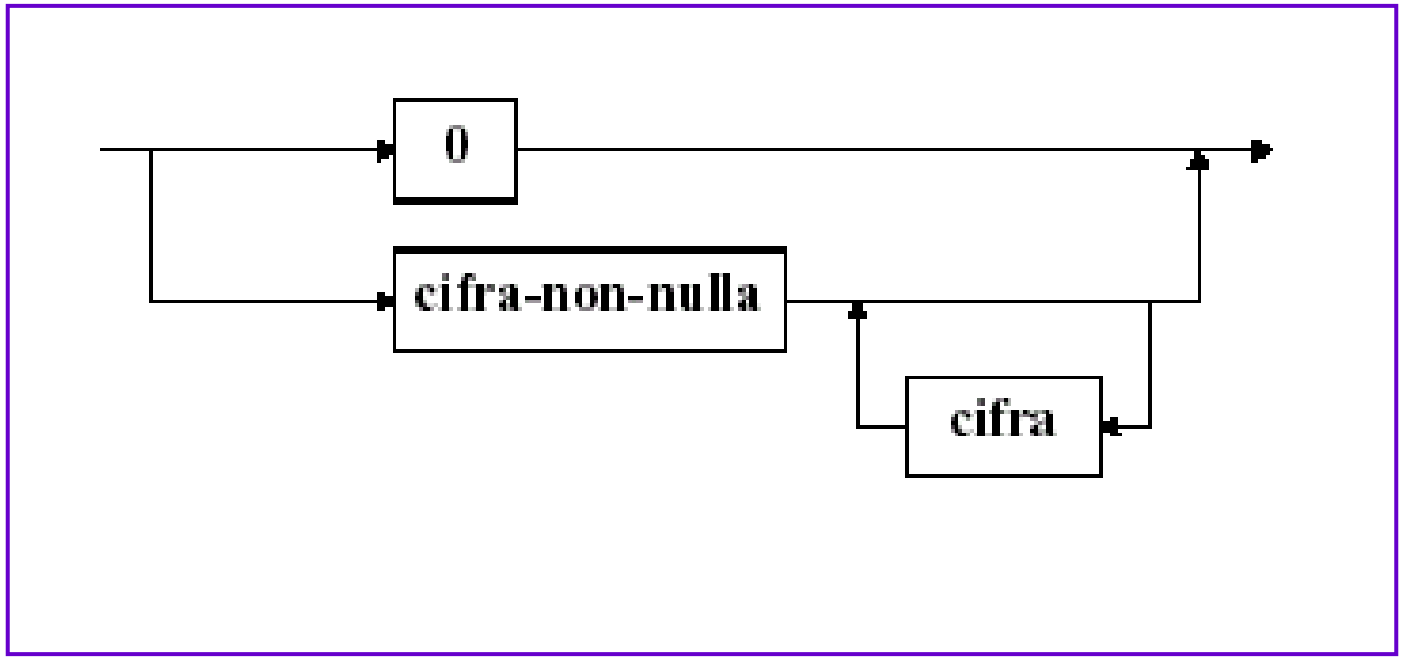
$$< cifra - non - nulla > ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

- Questa notazione significa che una cifra non nulla si può scrivere come 1 oppure 2 oppure 3…

$$< cifra >::= 0 \mid < cifra - non - nulla >$$

- Questa notazione indica che una cifra si può scrivere come 0 o come cifra non nulla.

Esempio: sintassi di un numero naturale



I linguaggi di programmazione: cenni storici (1)

- Benché siano macchine in grado di compiere operazioni complesse, i calcolatori devono essere “guidati” per mezzo di istruzioni appartenenti ad un linguaggio specifico e limitato, a loro comprensibile.
- Un linguaggio di programmazione è costituito, come ogni altro tipo di linguaggio, da un **alfabeto**, con cui viene costruito un insieme di **parole chiave** (il vocabolario) e da un insieme di **regole sintattiche** per l’uso corretto delle parole del linguaggio.
- A livello hardware, i calcolatori riconoscono solo comandi semplici, del tipo *copia un numero*, *addiziona due numeri, confronta due numeri.*

I linguaggi di programmazione: cenni storici (2)

- I comandi realizzati in hardware definiscono il **set di istruzioni macchina** e i programmi che li utilizzano direttamente sono i programmi in **linguaggio macchina**.
- In linguaggio macchina…
  - …ogni “operazione” richiede l’attivazione di numerose istruzioni base;
  - …qualunque entità, istruzioni, variabili, dati, è rappresentata da numeri binari: i programmi sono difficili da scrivere, leggere e manutenere.
- Il linguaggio macchina riflette l’organizzazione della macchina più che la natura del problema da risolvere.

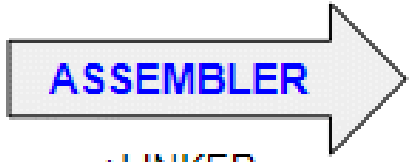
I linguaggi di programmazione: cenni storici (3)

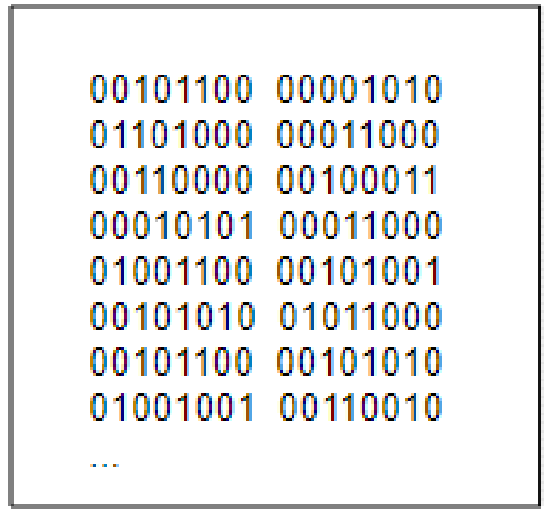
- Fino agli anni ‘70, tutti i programmi erano scritti in linguaggio macchina o in **assembly**.
- In assembly…
  - …ogni istruzione è identificata da una sigla piuttosto che da un codice numerico;
  - …il riferimento alle variabili viene effettuato per mezzo di nomi piuttosto che mediante indirizzi di memoria.
- I programmi scritti in assembly necessitano di un apposito programma **assemblatore** per tradurre le istruzioni tipiche del linguaggio in istruzioni macchina.

Esempio: un programma in assembly

```
;Scrivere un programma assembler che esegua la somma
;di 2 numeri (su 8 bit) che si trovano nelle locazioni
;di memoria OP1 e OP2, e ponga il risultato
;nella locazione RIS


        .MODEL SMALL
        .STACK
        .DATA
OP1     DB    9
OP2     DB    6
RIS     DB    ?
        .CODE
        .STARTUP
        MOV AL, OP1       ;SPOSTO IL 1° OPERANDO IN AL
        ADD AL, OP2       ;ESEGUO LA SOMMA
        MOV RIS, AL       ;MEMORIZZO IL RISULTATO
        .EXIT
        END
```

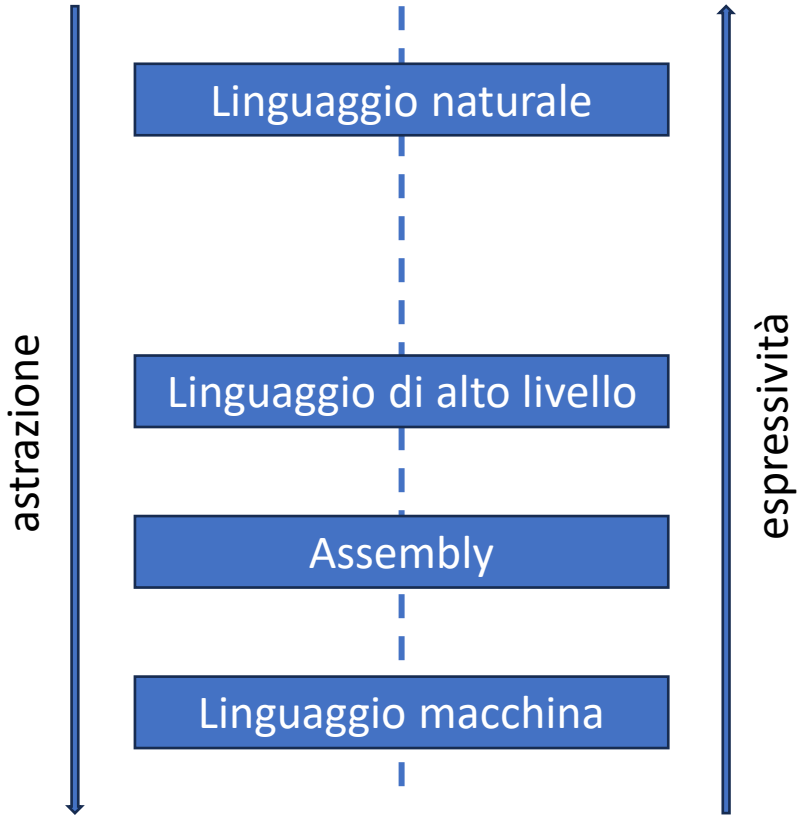




file oggetto eseguibile

I linguaggi di programmazione: cenni storici (4)

- Oggi si utilizza l'assembly solo se esistono vincoli stringenti sui tempi di esecuzione; viceversa, si usano linguaggi più vicini al linguaggio naturale, i **linguaggi di alto livello.**
- I linguaggi di alto livello sono elementi intermedi di una varietà di linguaggi ai cui estremi si trovano il linguaggio macchina, da un lato, ed i linguaggi naturali, come l'italiano e l'inglese, dall'altro.
- I linguaggi di programmazione differiscono comunque dai linguaggi naturali: sono infatti meno espressivi ma più precisi.
- Sono **semplici** e **poveri** (poche parole chiave, poche regole), ma **privi di qualsiasi ambiguità**.



Astrazione (1)

- In informatica si parla di **programmazione a basso livello** quando si utilizza un linguaggio molto vicino alla macchina.
- Si parla invece di **programmazione di alto livello** quando si utilizzano linguaggi più sofisticati ed astratti, slegati dal funzionamento fisico della macchina.
- Si viene così a creare una gerarchia di linguaggi, dai meno evoluti (il linguaggio macchina o l'assembly) ai più evoluti (Python, Java, etc.).

Astrazione (2)

- Esistono, quindi, diversi livelli di astrazione:
- **Linguaggio macchina e assembly**
  - Implicano la conoscenza dettagliata delle caratteristiche della macchina (registri, dimensione dati, set di istruzioni).
  - Semplici algoritmi implicano la specifica di molte istruzioni.
- **Linguaggi di alto livello**
  - Il programmatore può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico.
  - Sono indipendenti dalla macchina hardware sottostante.

Linguaggi di programmazione di alto livello (1)

- Consentono al programmatore di trattare oggetti complessi senza doversi preoccupare dei dettagli della particolare macchina sulla quale il programma viene eseguito.
- 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 traduttore simile ad un assemblatore, ma più complesso, infatti…
  - …esiste una corrispondenza biunivoca fra istruzioni in assembly ed istruzioni macchina;
  - …ogni singola istruzione di un linguaggio di alto livello corrisponde a molte istruzioni in linguaggio macchina: quanto più il linguaggio si discosta dal linguaggio macchina, tanto più il lavoro di traduzione del compilatore è difficile.

Linguaggi di programmazione di alto livello (2)

- I linguaggi che non dipendono dall'architettura della macchina offrono due vantaggi fondamentali:
  - i programmatori non devono cimentarsi con i dettagli architetturali di ogni calcolatore;
  - i programmi risultano più semplici da leggere e da modificare.
- Migliorano **portabilità**, **leggibilità** e **manutenibilità**.

Linguaggi di programmazione di alto livello (3)

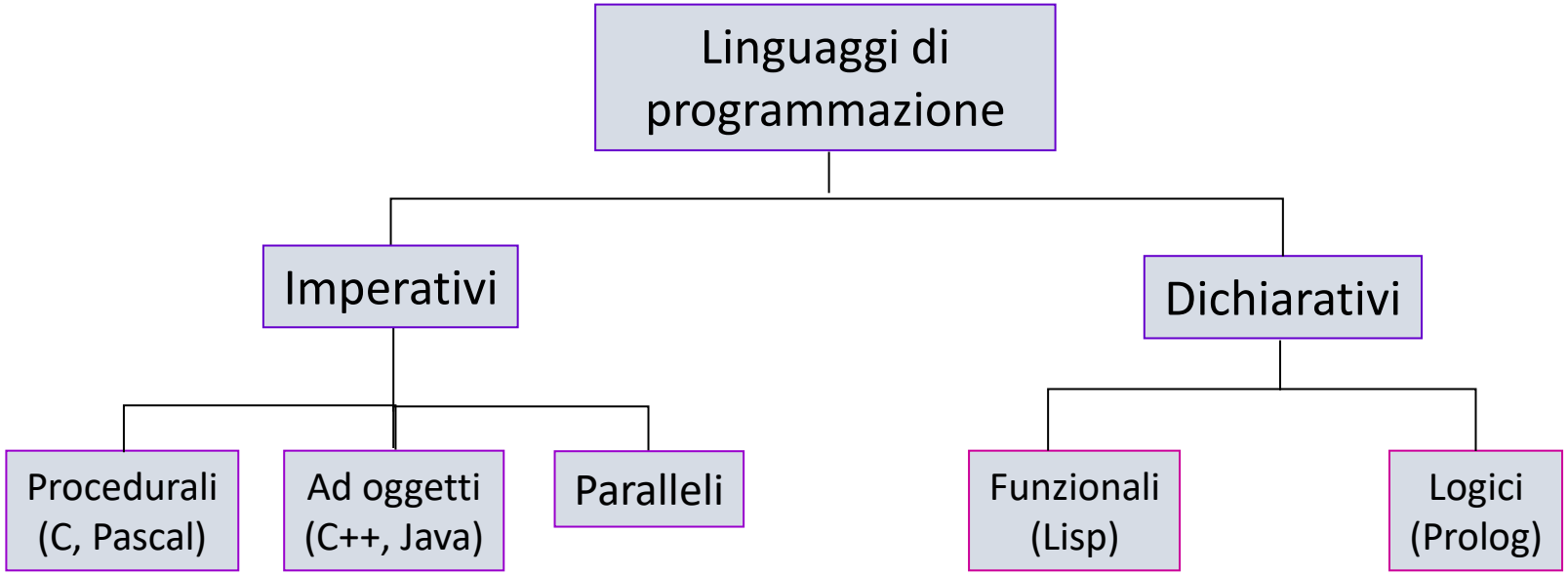
- **Portabilità**: i programmi scritti per un calcolatore possono essere utilizzati su qualsiasi altro calcolatore, previa ricompilazione.
- **Leggibilità**: la relativa similitudine con i linguaggi naturali rende i programmi più semplici, non solo da scrivere, ma anche da leggere.
- **Manutenibilità**: facilità nell'effettuare modifiche di tipo correttivo, perfettivo, evolutivo e adattivo.
- La possibilità di codificare algoritmi in maniera astratta si traduce in una migliore comprensibilità del codice e quindi in una più facile **analisi di correttezza**.

Linguaggi di programmazione di alto livello (4)

- Eventuale svantaggio dell'uso dei linguaggi di alto livello è la riduzione di efficienza.
  - *È possibile utilizzare successioni di istruzioni macchina diverse per scrivere programmi funzionalmente equivalenti: il programmatore ha un controllo limitato sulle modalità con cui il compilatore traduce il codice.*
  - *Tuttavia, compilatori sofisticati ricorrono a trucchi di cui molti programmatori ignorano l'esistenza.*
- La ragione fondamentale per decretare la superiorità dei linguaggi di alto livello consiste nel fatto che la maggior parte dei costi di produzione del software è localizzata nella fase di manutenzione, per la quale leggibilità e portabilità sono cruciali.

Tipi di linguaggi di programmazione di alto livello (1)

- Possiamo aggregare i numerosi linguaggi di programmazione esistenti sulla base del **modello astratto di programmazione** che sottintendono e che è necessario adottare per utilizzarli.



Tipi di linguaggi di programmazione di alto livello (2)

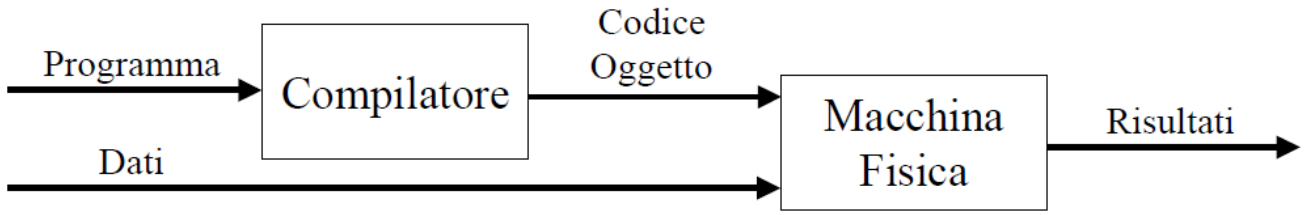
- **Linguaggi imperativi**
  - Il modello computazionale è basato sul **cambiamento di stato della memoria** della macchina.
  - È centrale il concetto di **assegnazione di un valore** ad una locazione di memoria (variabile).
  - Il compito del programmatore è costruire una sequenza di assegnazioni che producano lo stato finale (in modo tale che questo rappresenti la soluzione del problema).
- **Linguaggi dichiarativi**
  - Il modello computazionale è basato sui concetti di **funzione e relazione**.
  - Il programmatore non ragiona in termini di assegnazioni di valori, ma di relazioni tra entità e valori di una funzione.

Compilatori ed interpreti (1)

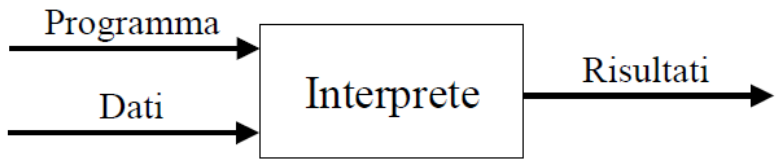
- Affinché un programma scritto in un qualsiasi linguaggio di programmazione sia comprensibile (e quindi eseguibile) da parte di un calcolatore, occorre tradurlo dal linguaggio originario al linguaggio della macchina.
- Ogni traduttore è in grado di comprendere e tradurre un solo linguaggio.
- Il traduttore converte il testo di un programma scritto in un particolare linguaggio di programmazione (**sorgente**) nella corrispondente rappresentazione in linguaggio macchina (programma **eseguibile**).

Compilatori ed interpreti (2)

- **Compilatore:** opera la traduzione di un programma sorgente (scritto in linguaggio di alto livello) in un programma oggetto direttamente eseguibile dal calcolatore.
  - *PRIMA si traduce tutto il programma*.
  - *POI si esegue la versione tradotta*.



- **Interprete:** traduce ed esegue il programma sorgente, istruzione per istruzione.
  - *Traduzione ed esecuzione sono intercalate.*



Compilatori ed interpreti (3)

Esempio di compilatore

- Dobbiamo sottoporre un curriculum, in inglese, ad una azienda, ma non conosciamo l'inglese.
- Abbiamo bisogno di un traduttore che traduca quanto scritto da noi dall'italiano all'inglese:
  - contattiamo il traduttore;
  - il traduttore riceve il testo da tradurre;
  - il traduttore fornisce il testo tradotto;
  - possiamo sottoporre il nostro curriculum all'azienda.

Compilatori ed interpreti (4)

Esempio di interprete

- Dobbiamo incontrare un manager cinese per motivi di lavoro ma non conosciamo il cinese.
- Abbiamo bisogno di un interprete che traduca il nostro dialogo:
  - contattiamo l'interprete;
  - parliamo in italiano, in presenza dell'interprete;
  - **contemporaneamente** l'interprete comunica al manager cinese quanto detto da noi e viceversa.
- Il compito dell'interprete si svolge contestualmente all'incontro col manager cinese.

Compilatori ed interpreti (5)

- Riassumendo…
  - I **compilatori** traducono un intero programma dal linguaggio *L* al linguaggio macchina della macchina prescelta.
    - *Traduzione ed esecuzione procedono separatamente.*
    - *Al termine della compilazione è disponibile la versione tradotta del programma.*
    - *La versione tradotta è però specifica per quella macchina.*
    - *Per eseguire il programma basta avere a disposizione la versione tradotta.*
  - Gli **interpreti** invece traducono e immediatamente eseguono il programma **istruzione per istruzione**.
    - *Traduzione ed esecuzione procedono insieme.*
    - *Al termine non vi è alcuna versione tradotta del programma originale.*
    - *Se si vuole rieseguire il programma occorre anche ritradurlo.*

Compilatori ed interpreti (6)

- L'esecuzione di un programma compilato è più veloce dell'esecuzione di un programma interpretato.
- I linguaggi interpretati sono tipicamente più flessibili e semplici da utilizzare (nei linguaggi compilati esistono maggiori limitazioni alla semantica dei costrutti).
- Per distribuire un programma interpretato si deve necessariamente distribuire il codice sorgente, rendendo possibili operazioni di plagio.
- Nei programmi interpretati, è facilitato il rilevamento di errori di run–time.

L’arte della programmazione (1)

- La soluzione di un problema tramite un programma è un procedimento che non si esaurisce nello scrivere codice in un dato linguaggio di programmazione, ma comprende una fase di progetto, che precede, e di verifica, che segue, la scrittura del codice.

- Definizione del problema
- Algoritmo per la soluzione del problema

← Analisi

- Codifica
- Debugging
- Validazione
- Documentazione
- Manutenzione

← Programmazione

L’arte della programmazione (2)

- **Definizione del problema**
  - Definizione degli ingressi e delle uscite
    - quali variabili
    - quale dominio per ogni variabile
  - Risoluzione delle ambiguità
  - Scomposizione in problemi più semplici
- **Definizione dell’algoritmo**
  - Soluzione in pseudocodice
  - Soluzione mediante diagramma a blocchi strutturato

L’arte della programmazione (3)

- **Codifica**
  - Traduzione dell’algoritmo in istruzioni del linguaggio di programmazione
- **Debugging:** correzione degli errori sintattici e semantici
  - Errori sintattici
    - Espressioni non valide o non ben formate nel linguaggio di programmazione
  - Errori semantici
    - Comportamento non aderente alle aspettative/alla intenzionalità del programmatore

L’arte della programmazione (4)

- **Validazione**
  - Test su tutte le condizioni operative del programma
  - Test su input estremi (es., vettori di dimensione 0 o 1, variabili nulle)
- **Documentazione**
  - Inserimento di commenti esplicativi nelle varie parti del programma per facilitarne la comprensione (dopo molto tempo dalla stesura o per terze persone)
- **Manutenzione**
  - Modifica del programma per soddisfare il cambiamento delle specifiche con cui deve operare

I commenti (1)

- Perché commentare e documentare i programmi?
  - I programmi vengono utilizzati più volte nel corso di tempi lunghi (mesi, anni) per…
    - …fare cambiamenti (aggiunta di caratteristiche);
    - …risolvere errori.
  - Commentare il programma serve a rendere chiaro ed evidente lo scopo delle varie parti del codice.

I commenti (2)

- Inoltre:
  - si devono evitare commenti inutili;
  - si deve evitare di inserirne “troppo pochi”.
- Un buon metodo per verificare il livello di documentazione è quello di leggere solo i commenti (e non il codice) ed ottenere una chiara idea su “cosa fa un programma e come lo fa”.

Domande?

42