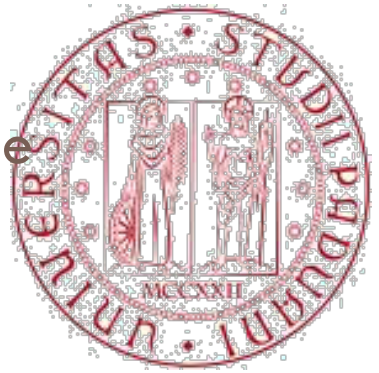
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

La programmazione

Prof.ssa Cinzia Pizzi

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Padova



Obiettivi

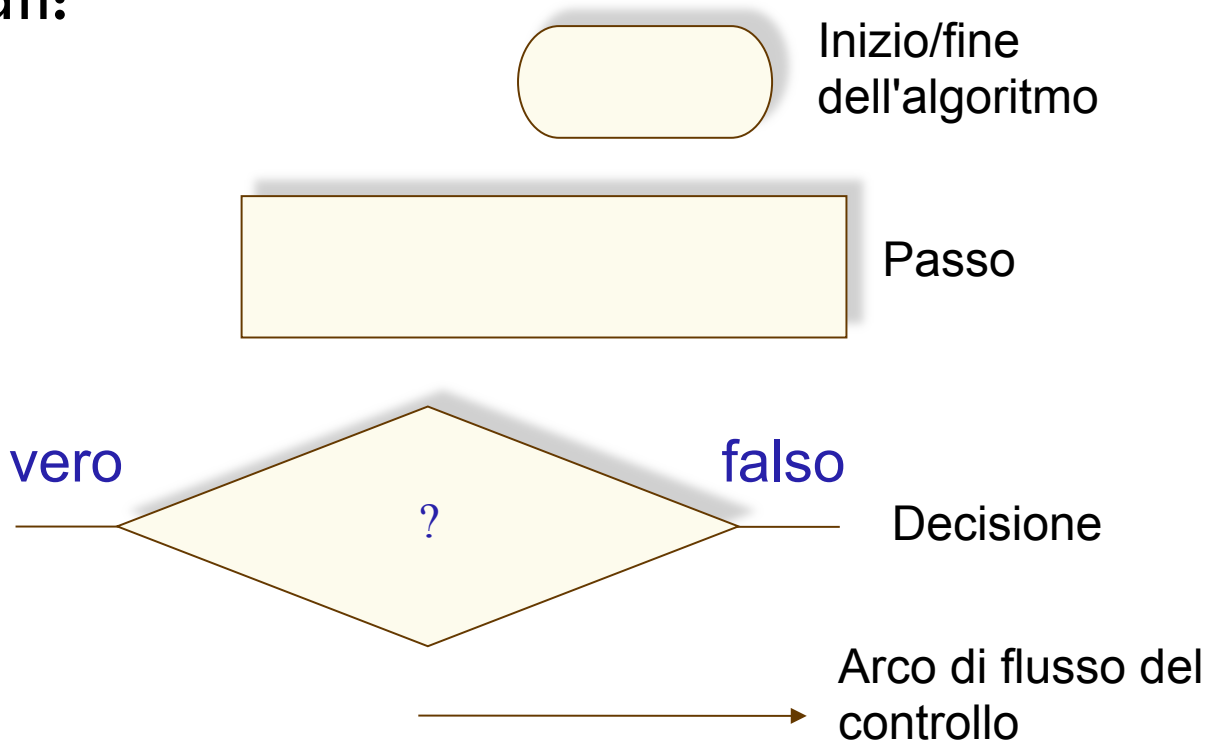
- Introduzione alla programmazione
  - Rappresentazione di un algoritmo
  - Definizione di programma
  - Correttezza
  - Tipi di istruzioni
- Tipi di linguaggi di programmazione
  - Linguaggi macchina e a basso livello
  - Linguaggi ad alto livello
    - Compilati, interpretati, ibridi

Come si descrive un algoritmo?

- Come si descrive un algoritmo?
  - In un linguaggio naturale
  - In linguaggio matematico
  - In un linguaggio di programmazione (vedremo!)
  - In un linguaggio specifico del contesto applicativo
    - Es. una costruzione grafica
  - In "pseudocodice", un linguaggio artificiale simile a molti linguaggi di programmazione
  - In una forma mista di tutte le precedenti

Diagrammi di flusso

- Semplici algoritmi sono a volte rappresentati graficamente per mezzo di diagrammi di flusso.
- Simboli usati:



Un esempio di algoritmo

- **Problema:** Avendo depositato ventimila euro in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo del conto arrivi al doppio della cifra iniziale?
- **Algoritmo:**

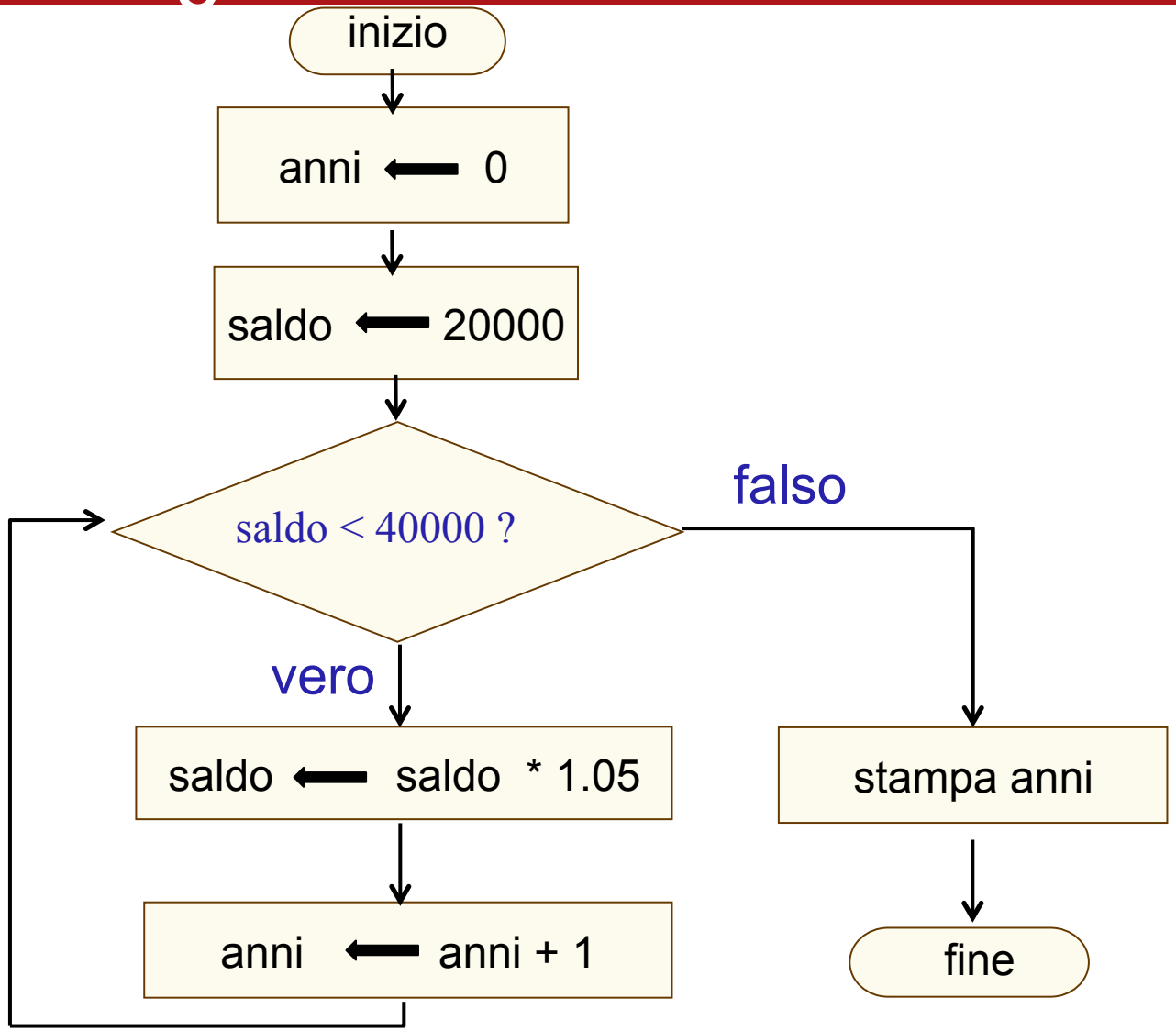
1. **Anni trascorsi** è 0 e il **saldo** è 20000€
2. Finché il **saldo** è minore di 40000€, ripetere i successivi passi **3** e **4**, poi passare al punto **5**
3. Aggiungere 1 al valore degli **anni trascorsi**
4. Il nuovo **saldo** è il valore attuale del **saldo** moltiplicato per 1.05 (cioè aggiungiamo il 5%)
5. Il risultato è il valore attuale degli **anni trascorsi**

anni	saldo
0	20.000,00
1	21.000,00
2	22.050,00
3	23.152,50
4	24.310,13
5	25.525,63
6	26.801,91
7	28.142,01
8	29.549,11
9	31.026,56
10	32.577,89
11	34.206,79
12	35.917,13
13	37.712,98
14	39.598,63
15	41.578,56

Un esempio di algoritmo

- Il metodo di soluzione proposto
  - **è non ambiguo**
    - fornisce precise istruzioni su cosa bisogna fare ad ogni passaggio e su quale deve essere il passaggio successivo
  - **è eseguibile**
    - ciascun passaggio può essere eseguito concretamente (se, ad esempio, il metodo di soluzione dicesse che il tasso di interesse da usare al punto **4** è variabile in dipendenza da fattori economici futuri, il metodo non sarebbe eseguibile…)
  - **arriva a conclusione in un tempo finito**
    - ad ogni passo il saldo aumenta di almeno mille euro, quindi al massimo in 20 passi arriva al termine

Rappresentazione dell'algoritmo con diagramma di flusso



Cos'e' la programmazione

- Un computer è una macchina che
  - ***memorizza dati*** (numeri, parole, immagini, suoni...)
  - ***interagisce con dispositivi*** (schermo, tastiera, mouse...)
  - ***esegue programmi***

- La ***programmazione*** è la disciplina che ha per oggetto la **definizione, progettazione** e **codifica** di programmi per calcolatore

Cosa sono i programmi

- I programmi sono implementazioni di algoritmi in un qualche linguaggio di programmazione
  - sequenze di istruzioni che il computer esegue e di decisioni che il computer prende per svolgere una certa attività

- Un programma è **corretto** se presenta
  - Correttezza sintattica
  - Correttezza semantica
  - Correttezza della soluzione

Com'è fatto un programma?

- Le istruzioni di cui sono composti i programmi sono ***molto elementari e di tre tipi (Bohm-Jacopini)***
  - Istruzioni imperative
    - leggere un numero da una posizione della memoria
    - sommare due numeri
    - accendere un punto rosso in una data posizione dello schermo
  - Istruzioni condizionali (che prendono ***decisioni***)
    - Es: se un dato è negativo, proseguire il programma con l'istruzione presente a un determinato indirizzo anziché con la successiva
  - Cicli (iterazioni): ripeti un blocco di istruzioni finche' una data condizione diventa falsa

Programmi e computer

- L'***elevatissimo numero*** di tali istruzioni presenti in un programma e la loro esecuzione ad ***altissima velocità*** garantisce l'illusione di una interazione fluida

- Il computer, in conclusione, è una macchina estremamente ***versatile***, caratteristica che le è conferita dai **molteplici programmi** che vi possono essere eseguiti, ciascuno dei quali consente di svolgere una determinata attività

Perche' imparare a programmare?

- Usare un computer non richiede alcuna capacità di programmazione
- La programmazione è necessaria per far fare al computer tutto (o quasi) quello che vogliamo noi
  - E un'attività che ***in genere*** affascina gli studenti e li motiva allo studio
- **Uno degli scopi di questo corso** è fornire la competenza per scrivere semplici programmi usando il linguaggio di programmazione Java

I linguaggi di programmazione

Il linguaggio del computer

- **Istruzioni macchina**: le istruzioni elementari eseguite da un computer (cioè dalla sua CPU)

- L'insieme di istruzioni macchina (***instruction set)*** è **specifico** di una particolare CPU

Le istruzioni macchina

- Esempio di descrizione di alcune istruzioni macchina:
  - carica in un registro il valore contenuto nella posizione di memoria 40
  - carica in un altro registro il valore 100
  - se il primo valore è maggiore del secondo, prosegui con l'istruzione contenuta nella posizione di memoria 240, altrimenti con l'istruzione che segue quella attuale

Le istruzioni macchina

- La codifica delle istruzioni macchina avviene sotto forma di ***configurazioni di bit*** conservate in memoria (che possono essere interpretate come numeri interi)

- Le precedenti istruzioni diventano

21 40 16 100 163 240

Categorie di istruzioni macchina

- ***trasferimento dati***, tra i registri e la memoria principale
  - LOAD (verso un registro), STORE (verso la memoria)
- ***operazioni aritmetiche e logiche***, eseguite dalla ALU
  - aritmetiche: ADD, SUB , MUL, DIV
  - logiche: AND, OR, NOT
- ***salti***, per alterare il flusso di esecuzione sequenziale (viene modificato il Program Counter)
  - Incondizionato (JUMP) : salta in ogni caso
  - condizionato: salta solo se un certo valore è zero (JZ) o se è maggiore di zero (JGZ)

Esecuzione di istruzioni macchina

- Per eseguire un programma in un computer è necessario ***scrivere all'interno della memoria primaria le configurazioni di bit corrispondenti alle istruzioni macchina del programma***

- Per fare ciò è necessario conoscere tutti i codici numerici delle istruzioni macchina

21 40 16 100 163 240...

L'assemblatore

```
iload      40
bipush    100
if_icmpgt 240
```

- Cos'è? Un programma
- Cosa fa?
  - Permette di scrivere il programma mediante dei ***nomi abbreviati*** (***codici mnemonici***) per le istruzioni macchina
    - più facili da ricordare per il programmatore
  - ***traduce*** il programma in configurazioni di bit
- I linguaggi espressi con codici mnemonici si dicono ***linguaggi assembly*** (uno diverso per ogni CPU)
  - **corrispondenza biunivoca** fra insieme di istruzioni assembly e instruction set del processore

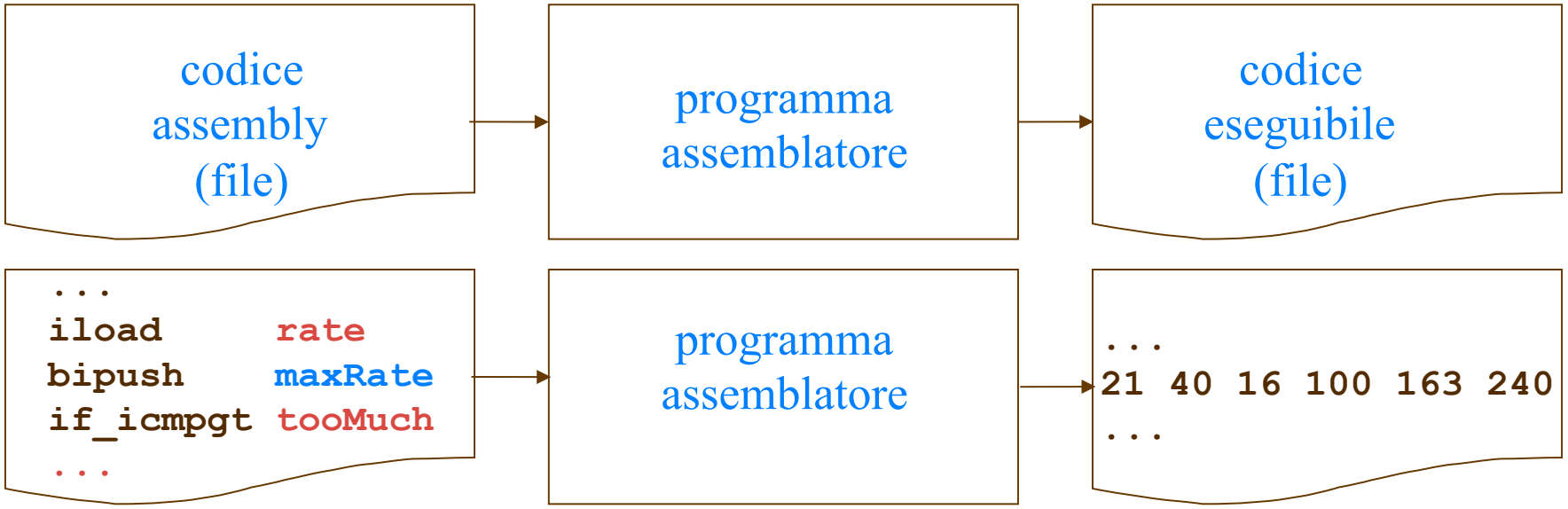
L'assemblatore

- Permette anche di assegnare dei ***nomi*** agli ***indirizzi di memoria*** e ai ***valori numerici*** e di usarli nelle istruzioni

```
iload      rate
bipush     maxRate
if_icmpgt  tooMuch
```

  - Programma ***molto più leggibile***
    - viene evidenziato il ***significato*** degli indirizzi di memoria e dei valori numerici

L'assemblatore



corrispondenza biunivoca tra

insieme di istruzioni assembly e instruction set della CPU

I linguaggi assembly

- **Vantaggi**
  - rappresentarono un grosso passo avanti rispetto alla programmazione in linguaggio macchina
- **Problemi**
  - occorrono ***molte istruzioni (1:1 con linguaggio macchina)*** per eseguire anche le operazioni più semplici
  - la sequenza di istruzioni di uno stesso programma ***cambia*** al cambiare della CPU
  - è molto costoso scrivere programmi che possano funzionare su diverse CPU

Linguaggi ad alto livello

- Il programmatore esprime la sequenza di operazioni da compiere, senza scendere al livello di dettaglio delle istruzioni macchina

```
if (rate > 100)
    System.out.print("Troppo");
```

Cos'è un linguaggio di programmazione?

- I linguaggi di programmazione sono
  - Linguaggi formali (artificiali)
  - Progettati per essere
    - Sufficientemente "espressivi" per descrivere un'ampia gamma di algoritmi
    - Comprensibili ai calcolatori in modo non ambiguo e possibilmente "efficiente"
    - Comprensibili a un programmatore umano in modo non ambiguo e possibilmente "naturale"
  - **Descritti da una grammatica**

Cos'è una grammatica?

- La grammatica di un linguaggio (non solo di programmazione) è un insieme di
  - Regole lessicali
    - come sono fatte le "parole" del linguaggio?
  - Regole sintattiche
    - come sono fatte le "frasi" del linguaggio?
  - Regole semantiche
    - che significato hanno le frasi del linguaggio?

Come si descrive una grammatica?

- La grammatica di un linguaggio di programmazione viene solitamente descritta usando una notazione particolare
  - EBNF – Extended Backus-Naur Form
    - John Backus e Peter Naur la introdussero per descrivere il linguaggio di programmazione Algol
  - EBNF è un linguaggio per descrivere linguaggi!
- In pratica in questo corso non useremo (E)BNF per descrivere Java, ma è bene almeno sapere che esiste

Esempio

(a) Syntax rules

<program> : :={ <statement>* }

<statement> : :=<assignment> | <conditional> | <loop>

<assignment> : :=<identifier> := <expr>;

<conditional> : :=**if** <expr> { <statement>$^+$ } |

if <expr> { <statement>$^+$ } **else** { <statement>$^+$ }

<loop> : :=**while** <expr> { <statement>$^+$ }

<expr> : :=<identifier> | <number> | (<expr>) |

<expr> <operator> <expr>

(b) Lexical rules

<operator>::= **+** | **-** | * | **/** | **=** | ≠ | **<** | **>** | ≤ | ≥

<identifier>::= < letter> <Id>*

<Id>::= < letter> | <digit>

<number>::= <digit>$^+$

<letter>::= a | b | c | … | z

<digit>::= 0 | 1 | … | 9

FIGURE 2.1 EBNF definition of a simple programming language

Un programma e' una sequenza di istruzioni

Una istruzione e' un'assegnamento, un quesito condizionale o un ciclo

Linguaggi ad alto livello

- I linguaggi ad alto livello possono essere distinti in
  - Linguaggi compilati
  - Linguaggi interpretati

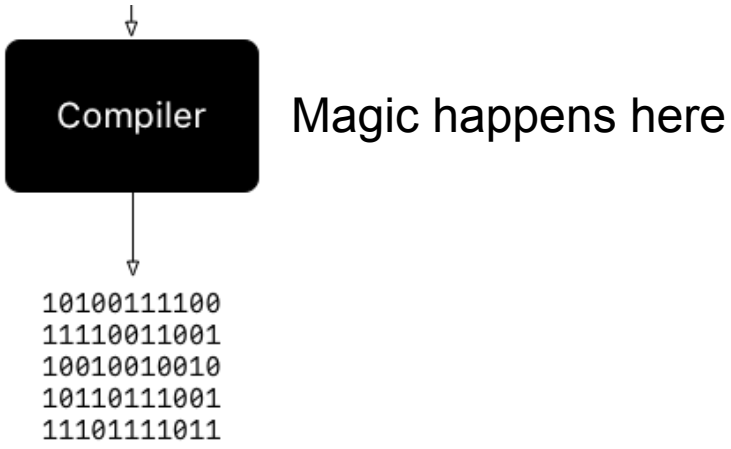
Linguaggi compilati

- Il **compilatore è un programma che**
  - legge il programma in linguaggio ad alto livello
  - genera il corrispondente programma nel linguaggio macchina di una certa CPU
- Dipendenze
  - I linguaggi sono indipendenti dalla CPU
  - Il prodotto della compilazione (codice eseguibile), non è indipendente dalla CPU
  - occorre compilare il programma con un diverso compilatore per ogni CPU sulla quale lo si vuole eseguire

Linguaggi compilati

codice sorgente: codice scritto in un linguaggio ad alto livello

```
void stampa_saluto{
      printf("%s","Benvenuto");
}
```



codice eseguibile: codice scritto nel linguaggio macchina

Linguaggi interpretati

- ***L'interprete*** è un programma che:
  - traduce "al volo" il codice sorgente in codice eseguibile e lo esegue, **senza creare un file eseguibile**

- Ogni volta che si vuole far eseguire il programma è necessario che l'interprete **legga – traduca – esegua** il programma, istruzione dopo istruzione

Linguaggi ad alto livello

- Negli anni '50 furono inventati i primi linguaggi di programmazione ***ad alto livello***
  - **FORTRAN**: primo "vero" linguaggio, **BASIC**, **COBOL**
  - **Presenza di "salti" (spaghetti code)**
- Anni '60 e '70: programmazione strutturata
  - **Pascal** (Niklaus Wirth, 1968), **C** (B. Kernigham e D. Ritchie, 1970-75)
  - Strutture di controllo: sequenza-selezione-iterazione
  - Un punto di ingresso, uno di uscita: componibilità
- Anni '80 e '90, programmazione orientata agli oggetti
  - **C++** (B. Stroustrup, 1979) , **Java** (J.Gosling e P. Naughton, 1991)
  - Definisce "classi": modelli astratti che descrivono dati e operazioni su esse (metodi). Gli oggetti sono istanze di una classe.

Linguaggi ad alto livello

- I linguaggi di programmazione, creati dall'uomo, hanno una sintassi molto più rigida di quella dei linguaggi naturali, per agevolare il compilatore
- Il compilatore segnala gli ***errori di sintassi***

```
if (rate > 100)) System.out.print("Troppo");
```

C'è una parentesi chiusa di troppo

- e ***non tenta di capire***, come farebbe un utente umano, perche' sarebbe molto difficile verificare le ***intuizioni*** del compilatore
  - ***meglio la segnalazione di errori!***

Linguaggi ad alto livello

- Esistono molti linguaggi di programmazione ad alto livello, ognuno con la propria sintassi
- L'esempio seguente è in linguaggio Java

  if (rate > 100) System.out.print("Troppo");
- e questo è l'equivalente in linguaggio Pascal

  if rate > 100 then write ('Troppo');
- La sintassi è un compromesso tra
  - Leggibilità
  - Facilità di compilazione
  - Coerenza con altri linguaggi

Riassunto

- Linguaggi Macchina
  - Il computer capisce sequenze binarie 0011001101
  - Posso rappresentare le istruzioni con dei numeri
    21 40 16 100 163 240
  - Difficile ricordare i codici delle istruzioni
  - Ogni CPU ha il proprio **instruction set**
- Linguaggio Assembly
  - Associazione 1:1 tra instruction set e codici mnemonici
  - Piu' facili da ricordare, maggior leggibilita' del codice
  - Ancora dipendente dalla CPU

Riassunto

- Linguaggi ad alto livello
  - Il programmatore esprime la sequenza di operazioni da compiere, senza scendere al livello di dettaglio delle istruzioni macchina
  - La traduzione in linguaggio macchina viene fatta da un altro programma (compilatore e/o interprete)
    - La sintassi è molto rigida, non può esserci ambiguità
    - Maggior semplicità nella scrittura del programma
    - Maggior indipendenza dall'instruction set della CPU

Introduzione a Java

Il linguaggio Java

- Nato nel 1991 in Sun Microsystems, da un gruppo di progettisti guidato da Gosling e Naughton

- Progettato per essere ***semplice*** e ***indipendente dalla CPU***

- Il primo prodotto del progetto Java fu un ***browser***, **HotJava**, presentato nel 1994, che poteva scaricare programmi (detti ***applet***) da un server ed eseguirli sul computer dell'utente, ***indipendentemente*** dalla sua piattaforma

Il linguaggio Java

- Il linguaggio Java è stato adottato da moltissimi programmatori perché
  - E' più ***semplice*** del suo diretto concorrente, C++
  - consente di ***scrivere e compilare una volta ed eseguire dovunque*** (cioè su tutte le piattaforme) **"compile once, execute everywhere"**
    - Il compilatore non da in uscita codice macchina ma un codice intermedio (**bytecode**) che poi viene interpretato
  - ha una **ricchissima libreria standard** che mette a disposizione dei programmatori un insieme vastissimo di funzionalità ***standard***, indipendenti anche dal sistema operativo

La libreria di classi standard

- Per effettuare molte operazioni comuni ad esempio: scrivere sulla shell ("standard output")
  - è necessario ***interagire con il sistema operativo*** operazioni ***di basso livello*** che richiedono conoscenze specifiche
- Queste operazioni, per chi utilizza Java, sono state già realizzate dagli autori del linguaggio, che hanno scritto delle classi apposite (ad esempio, **System**)
- Il bytecode di queste classi si trova all'interno di ***librerie standard***, che sono ***raccolte di classi***
- ***Non è necessario avere a disposizione il codice sorgente di queste classi, né capirlo!*** Comodo...

Il linguaggio Java per gli studenti

- Dato che, diversamente da altri linguaggi di programmazione, **Java non è stato progettato per la didattica…**
  - ***non è molto semplice scrivere programmi Java molto semplici!***
- Anche per scrivere programmi molto semplici è necessario conoscere parecchi dettagli "tecnici", un potenziale problema nelle prime lezioni
- Adotteremo lo stile didattico di ***usare alcuni costrutti sintattici senza spiegarli o approfondirli***, rimandando tali fasi al seguito

Java Development Kit (JDK)

- Da **scaricare** e **installare** sul vostro computer

- Il **Java Standard Editon Development Kit (JDK)** scaricabile dal sito
  https://www.oracle.com/java/technologies/downloads/

  contiene compilatore e interprete standard, cui faremo riferimento

Le fasi della programmazione

- In generale, con i linguaggi compilati, l'attività di programmazione si esegue in tre fasi:
  - scrittura del programma (**codice sorgente**)
  - compilazione del codice sorgente
    - creazione del **codice eseguibile** (codice macchina)
  - esecuzione del programma

Scrittura di un programma

- Per scrivere il codice sorgente si usa un ***editor di testo***, *salvando* (memorizzando) il codice (detto codice sorgente) in un file

- Nel linguaggio Java il file deve avere estensione .java, ad esempio Hello.java

Compilazione del codice sorgente in Java

- Compiliamo il codice sorgente di un programma Java
  - **javac** è il comando per compilare un file con estensione .java
    - Es: javac Hello.java

- Otteniamo un particolare formato di ***codice eseguibile***, detto ***bytecode***, che è ``codice macchina'' per la Java Virtual Machine (JVM)

Java Virtual Machine

- ***Macchina virtuale Java*** (JVM - *Java Virtual Machine*)
  - è un programma che ha lo scopo di eseguire altri programmi (è sostanzialmente un interprete che emula il funzionamento di un processore)
  - Permette di eseguire programmi java su qualsiasi dispositivo o sistema operativo
  - Gestisce e ottimizza l'uso della memoria

La compilazione del codice sorgente in Java

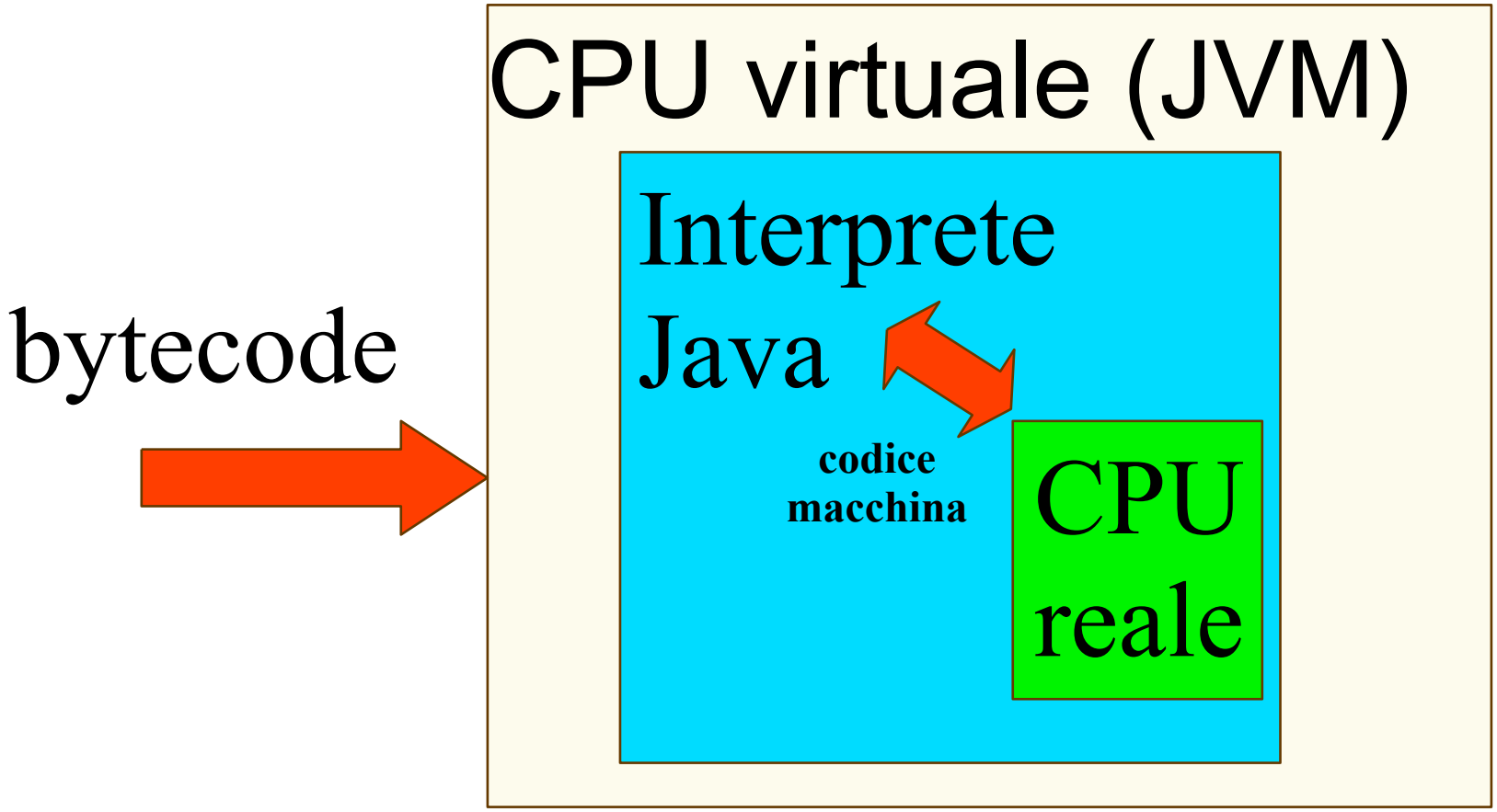
- L'istruzione **javac Hello.java** genera il file **Hello.class**

- Il file .class contiene il bytecode che non è codice direttamente eseguibile
  - Il file con il codice bytecode contiene una ***traduzione*** delle istruzioni del programma in codice interpretabile dalla JVM

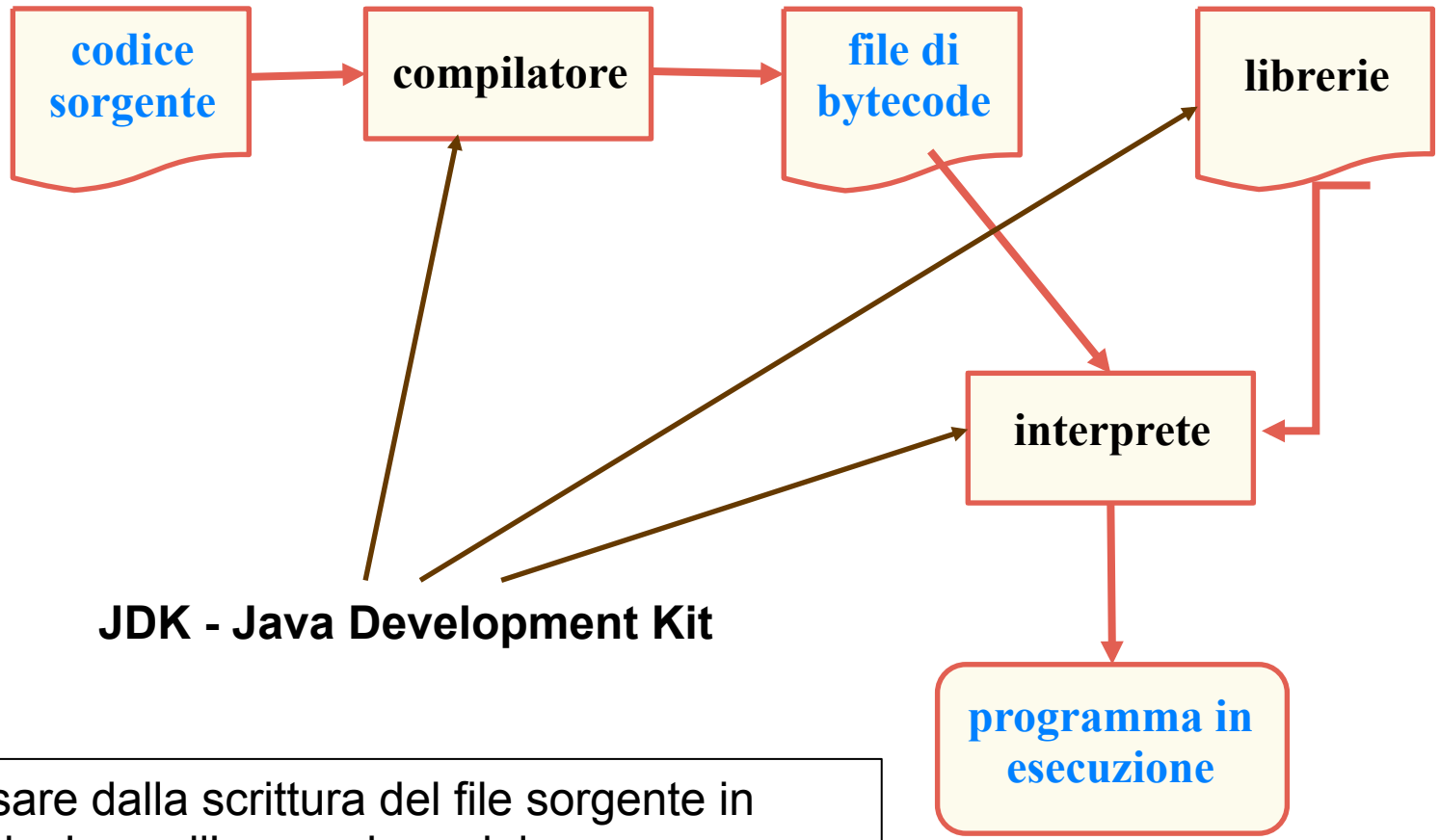
Esecuzione del programma

- Per eseguire un programma si usa l'**interprete Java** (comando **java**), un programma eseguibile sul computer dell'utente che
  - carica in memoria principale (RAM) il bytecode del programma (**Hello**)
  - avvia il programma
  - carica successivamente altri file di bytecode che sono eventualmente necessari durante l'esecuzione (per esempio dalla libreria standard)
  - traduce **"al volo"** le istruzioni del bytecode in istruzioni macchina della CPU reale (che esegue il codice)
- L'istruzione **java Hello** esegue il programma **Hello**

La Java Virtual Machine



Schema di processo di programmazione in Java

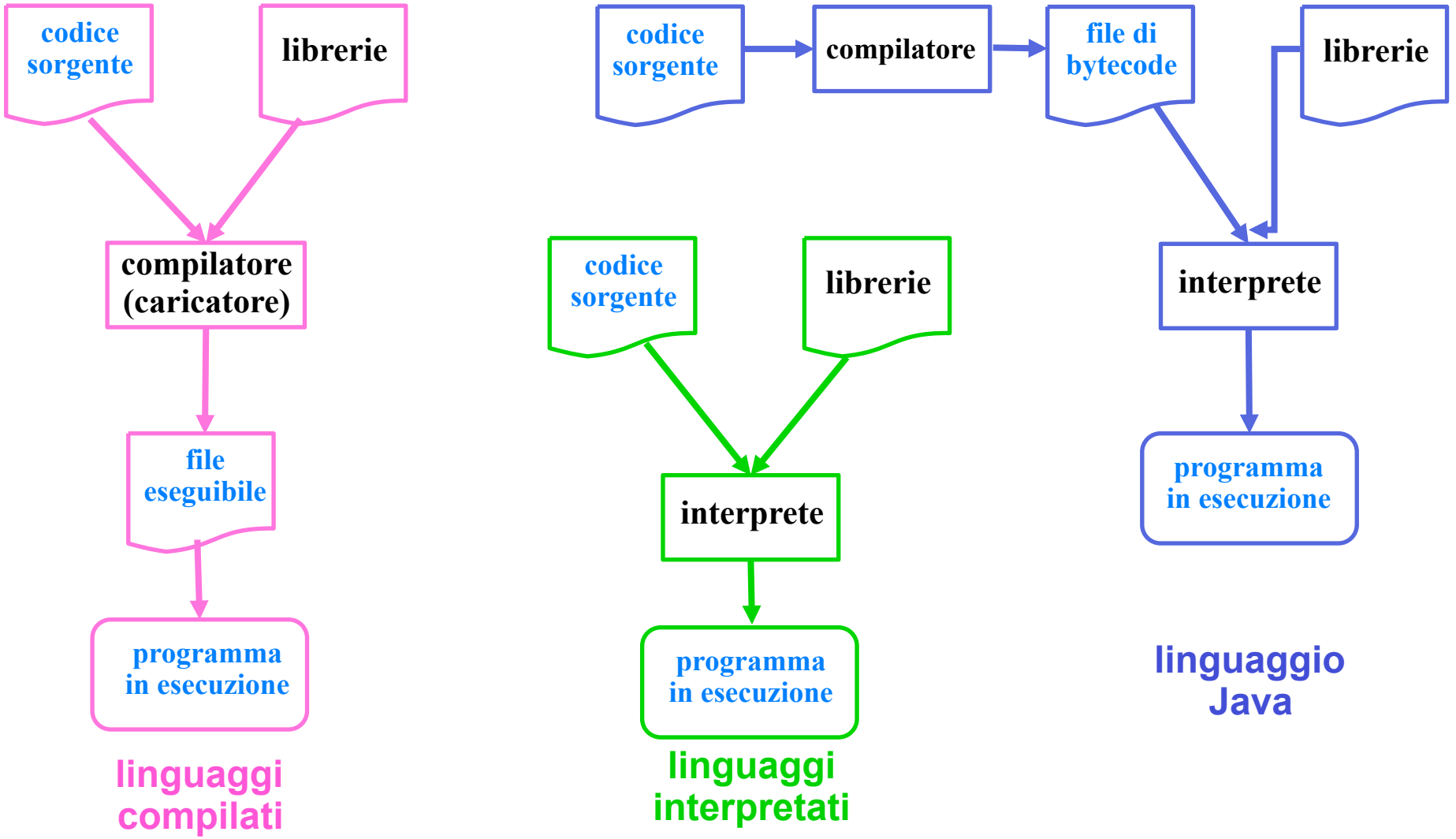


Per passare dalla scrittura del file sorgente in linguaggio Java all'esecuzione del programma su una particolare CPU, si usano il ***compilatore*** e l'***interprete***

Compilatore e/o inteprete

- La maggior parte degli altri linguaggi di programmazione ad alto livello, invece, usa soltanto il ***compilatore* oppure** soltanto l'***interprete***
  - con ***linguaggi compilati*** come C e C++, si usa il ***compilatore*** per creare un file eseguibile, contenente codice macchina, a partire da file sorgenti (con l'eventuale ausilio di un ***caricatore***, *loader*, che interagisce con una libreria)
  - con ***linguaggi interpretati*** come PERL e PYTHON, l'***interprete*** traduce "al volo" il file sorgente in codice eseguibile e lo esegue, senza creare un file eseguibile

Il processo di programmazione



Compilatore e/o inteprete

- Il fatto che un linguaggio sia compilato o interpretato influisce fortemente su
  - quanto è facile eseguire lo stesso programma su computer aventi diverse CPU (***portabilità***)
  - velocità nell'esecuzione di un programma (***efficienza)***

- *Il linguaggio Java, da questo punto di vista, è un* ***linguaggio misto****, essendo* ***sia compilato sia interpretato****, in fasi diverse*

Portabilità

- I programmi scritti in un linguaggio **interpretato** sono portabili
- I programmi scritti in un linguaggio **compilato**
  - sono portabili a livello di file sorgente, ma è necessario compilare il programma su ogni diversa CPU
  - non sono portabili a livello di file eseguibile, perché esso contiene codice macchina per una particolare CPU
- ***I programmi scritti in linguaggio Java sono portabili***, oltre che a livello di file sorgente, anche ***a un livello intermedio***, il livello del bytecode
  - possono essere compilati una sola volta ed eseguiti da ***interpreti diversi*** su diverse CPU

Efficienza

- I programmi scritti in un linguaggio interpretato sono poco efficienti
  - l'intero processo di traduzione in linguaggio macchina deve essere svolto a ogni esecuzione
- I programmi scritti in un linguaggio compilato sono molto efficienti
  - l'intero processo di traduzione in linguaggio macchina viene svolto prima dell'esecuzione, una volta per tutte
- I programmi scritti in linguaggio **Java** hanno un'**efficienza intermedia**
  - parte del processo di traduzione viene svolto una volta per tutte (dal compilatore) e parte viene svolto a ogni esecuzione (dall'interprete)

Portabilità vs efficienza

- Se si vuole soltanto la portabilità, i linguaggi interpretati sono la scelta migliore
- Se si vuole soltanto l'efficienza, i linguaggi compilati sono la scelta migliore
- Se si vogliono perseguire ***entrambi gli obiettivi***, come quasi sempre succede, ***il linguaggio Java può essere la scelta vincente***

Inoltre…

- Java
  - È largamente utilizzato
  - Java è facilmente disponibile
  - Costantemente in sviluppo dagli anni '90

- Fatti della vita…
  - Nessun linguaggio di programmazione è perfetto
  - Da qualche parte bisogna iniziare

Approccio di questo corso

- Usiamo solo un set minimale di Java
- Sviluppiamo delle abilità di programmazione generali che possono essere applicate a molti altri linguaggi

Take home message

- Il linguaggi di programmazione sono un compromesso tra il linguaggio naturale e il linguaggio macchina
- Diversi tipi di linguaggi di programmazione
  - Compilati (C/C++)
  - Interpretati (perl, python)
  - Ibridi (Java)
- Java: molto utilizzato, buon compromesso tra portabilità ed efficienza
  - Obiettivo: sviluppare skill di programmazione indipendentemente dal linguaggio!