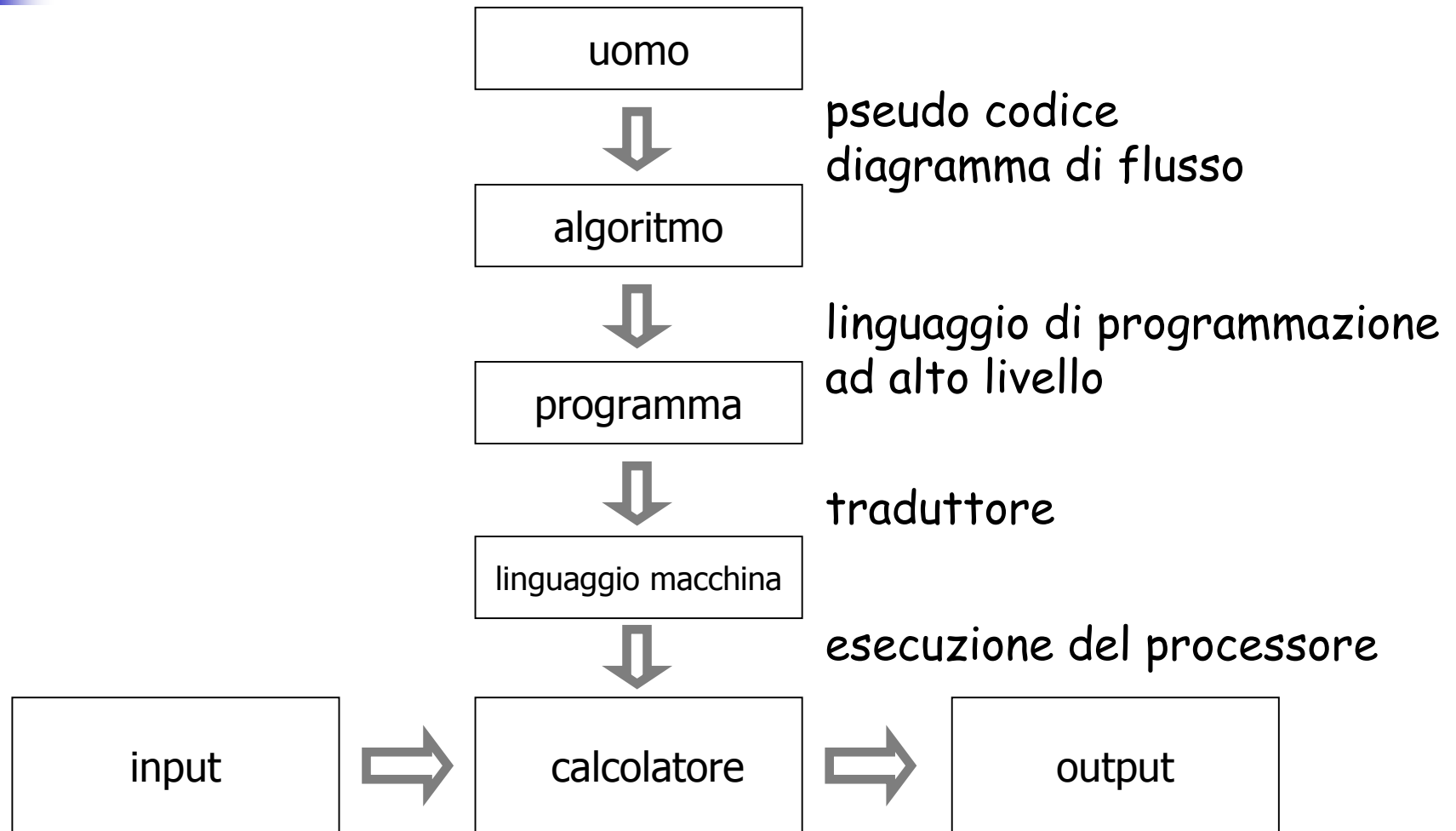




Cosa è un programma

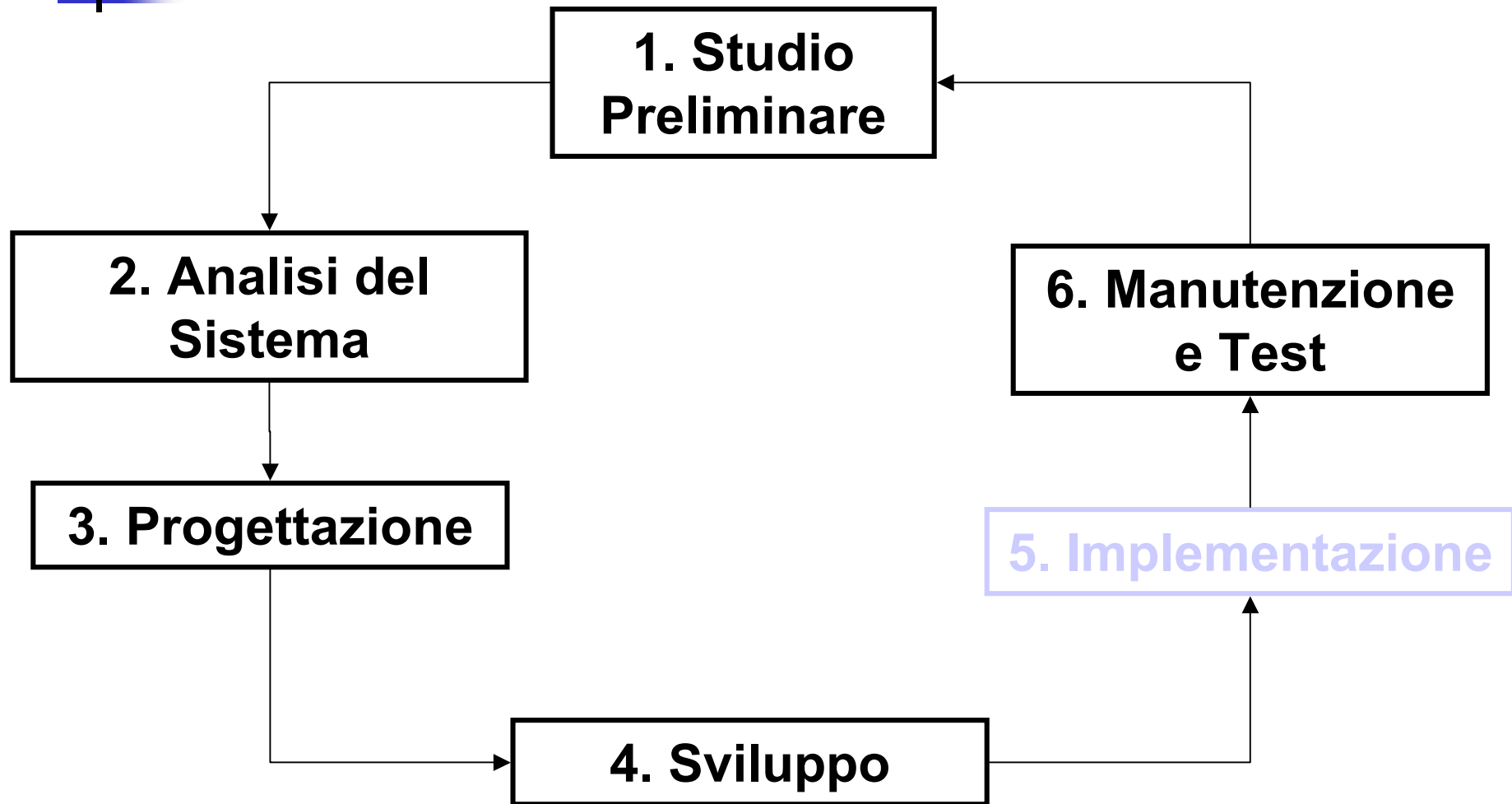
- Il *programma* è la "scatola nera" che risolve il problema computazionale;
- Il programma è una sequenza di istruzioni che devono essere eseguite;
- Il programma è la traduzione per il computer dei passi dell'algoritmo
- Deve essere scritto tramite un linguaggio che il computer capisca (*linguaggio macchina*);
- Il programmatore utilizza un *linguaggio di programmazione* che poi viene tradotto (da qualcosa) in linguaggio macchina;

Dall'uomo al calcolatore





Fasi di creazione di un programma





Le fasi

- Analisi del problema computazionale (individuare input ed output, etc.)
- Progettazione di una soluzione proponendo una suddivisione del problema in sottoproblemi più semplici; alla fine di questa fase, per ogni sottoproblema si deve scrivere l'*algoritmo* utilizzando *pseudo-codici* e *diagrammi di flusso*;
- Stesura del programma a partire dall'algoritmo: **implementazione** vera e propria
- Test del programma: alfa testing, beta testing, debugging
- **Produzione documentazione e distribuzione**



Un altro esempio di algoritmo

Preparazione del **Risotto alla Zucca** da parte di un esecutore in grado di aggiungere, tagliare, mescolare, tostare, mantecare,...

1. Preparare il soffritto ed il brodo;
2. Aggiungere la zucca a pezzettini al soffritto;
3. Mescolare **fino a quando** la zucca non è un purè;
4. Aggiungere il riso al soffritto;
5. Fare tostare il riso; poi bagnarlo con il vino;
6. Aggiungere brodo **fino a quando** il riso è cotto;
7. Aggiungere prezzemolo, pepe e burro;
8. **Se** preferisci salato, **allora** aggiungi sale;



Sottoproblemi

- L'algoritmo precedente è un altro esempio di cosa significa scomporre un **problema in sottoproblemi**;
 - Come si prepara un soffritto?
 - Come si prepara un brodo?
- Ogni sottoproblema può essere scomposto in problemi via via più *elementari*;
- Per ogni problema elementare deve esistere un'istruzione nel linguaggio adottato la cui esecuzione lo risolve.



Alto e basso livello

- Man mano che si suddivide il problema in sottoproblemi, si scende di livello;
- Se ci si avvicina al linguaggio umano, si parla di linguaggi di *Alto livello*;
- Se ci si avvicina al linguaggio macchina, si parla di linguaggi di *Basso livello*;



Linguaggi

Linguaggio umano (es. Italiano)

Descrizione di un *algoritmo* (es. diagrammi di flusso,
pseudo codice, etc...)

Linguaggi di Programmazione
(es. C, VBasic, Java, Logo, etc...)

Alto livello

Linguaggio Macchina

Basso livello



Linguaggi di programmazione

- Linguaggio macchina
 - Codifica binaria delle istruzioni
 - Diverso per ogni architettura
 - *Istruzioni e operazioni elementari coincidono*
- Linguaggio Assembler
 - Sempre a livello macchina, ma codifica simbolica delle istruzioni
 - Es: ADD R1, R6
STORE R1, RAM[255]
MOV R1, #4



Linguaggi di programmazione

- Linguaggio ad alto livello (HLL - High-Level Languages)
 - Elaboratore virtuale
 - Operazioni molto più astratte delle operazioni HW
 - Es: $x = y + 2$
 $z = \cos(x)$
 - Linguaggio indipendente dalla piattaforma!
 - Possibile mediante l'esistenza di traduttori

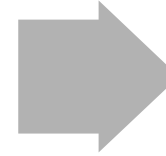


Linguaggi di programmazione

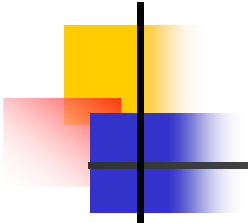
*Programma
sorgente
(HLL)*



Traduttore



*Programma
eseguibile
(bit)*



Traduttori



“somma due interi”
...
“stampa un
messaggio”
...

Uomo

Traduttore

010101010111
010101111111
101010101010
101010101011



Macchina



Traduttori

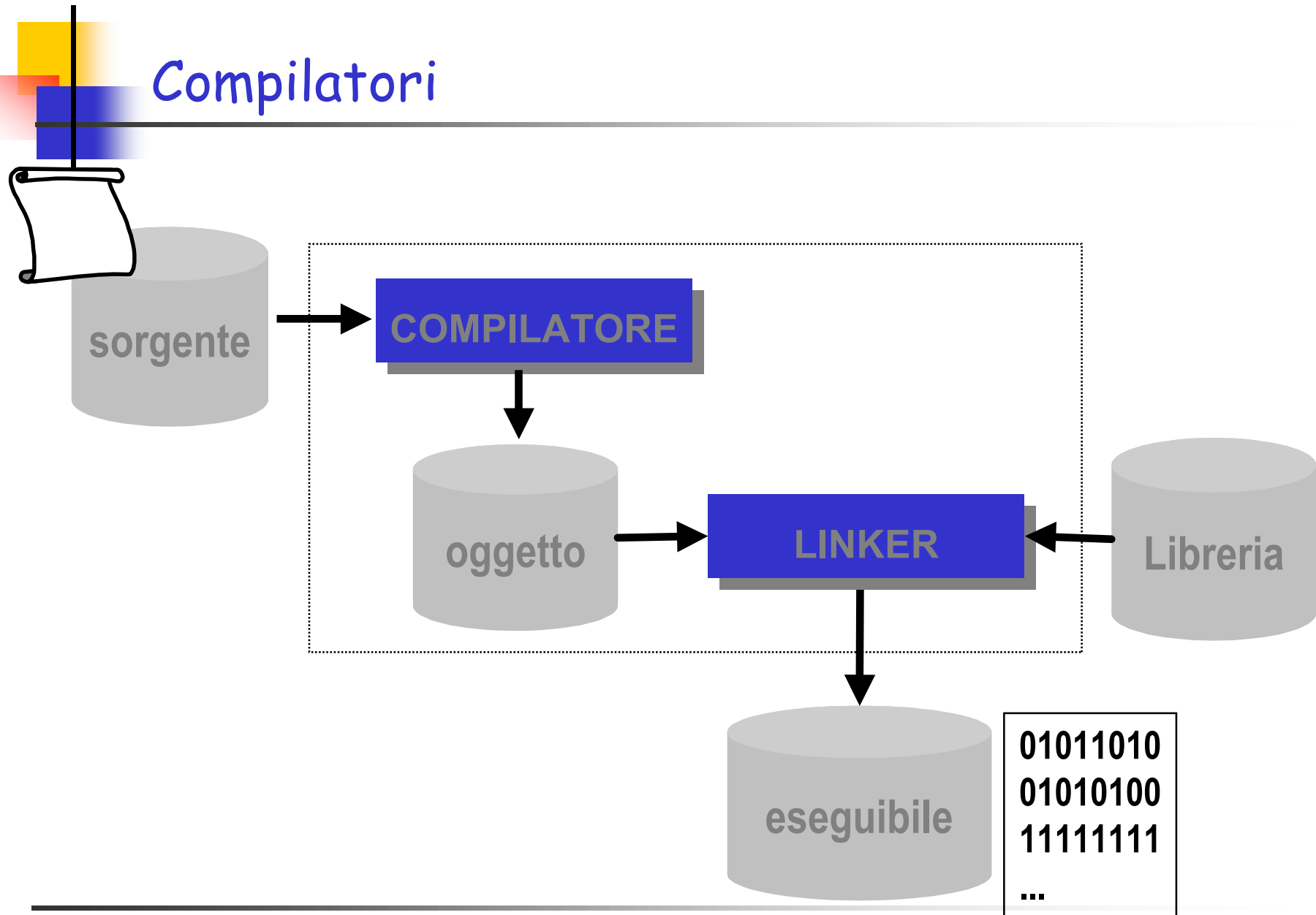
- Due schemi
 - Compilatori
 - Interpreti



Compilatori

- Traduzione avviene in due passi:
 - Compilazione vera e propria
 - Collegamento (link)
- Dopo la fase di link, il programma può essere eseguito direttamente
- Ogni fase produce un file corrispondente all'aggiunta di informazioni di vario tipo
 - Formato oggetto
 - Formato eseguibile

Compilatori





Linker

- Risolve i riferimenti ad indirizzi di memoria
 - Indirizzi logici => indirizzi fisici
- Aggiunge al codice le librerie:
 - Funzioni pre-compilate e riutilizzabili (es. funzioni matematiche) e distribuite con il compilatore
 - Codice scritto in precedenza
 - Due schemi:
 - Librerie statiche
 - Librerie dinamiche



Interpreti

- Il formato interno viene generato interpretando e traducendo il formato sorgente
- Il programma viene interpretato ed immediatamente eseguito *istruzione per istruzione*.
- Non viene generato né formato intermedio (oggetto) né eseguibile.

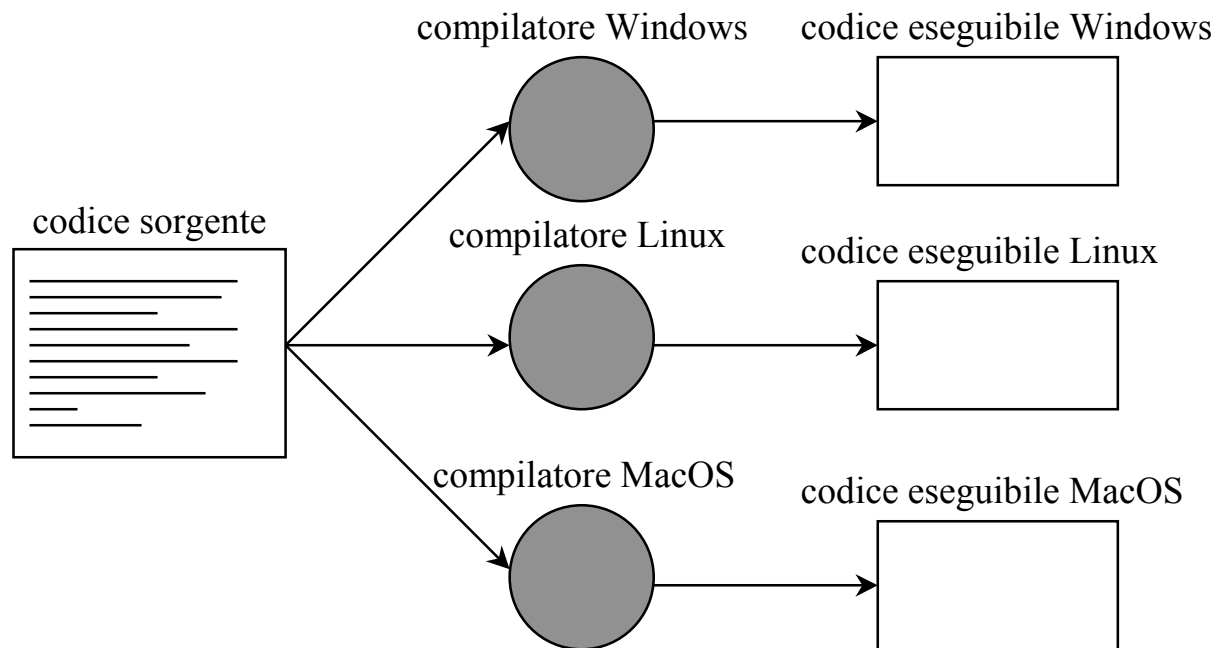


Compilatori vs. interpreti

- Linguaggi compilati:
 - Vantaggi:
 - Formato del programma eseguibile più efficiente → esecuzione più veloce
 - Migliore supporto per istruzioni "complesse" e programmi di grandi dimensioni
 - Svantaggi:
 - Rallentamento tempo di sviluppo dei programmi
 - Maggiori requisiti di spazio di memoria (il compilatore è un programma sofisticato)
- Linguaggi interpretati:
 - Ideali in ambiente didattico per lo sviluppo rapido di programmi.

Portabilità del software

- *Possibilità per un programma di poter essere eseguito su piattaforme HW e SW diverse da quelle su cui è stato sviluppato*





Linguaggi di programmazione

- **FORTRAN**
 - FORMula TRANslation (1956)
 - calcoli tecnico-scientifici (ambiente fisico/matematico)
 - svariate librerie
 - compilato
- **COBOL**
 - COmmerce and Business Oriented Language (1960)
 - elaborazione di archivi, tabulati
 - applicazioni contabili



Linguaggi di programmazione

- **BASIC**
 - Beginner's All-purpose Symbolic Instruction Code (1962)
 - relativamente semplice
 - capacità grafiche
 - interpretato
 - versioni "evolute" (VisualBasic)
- **PASCAL**
 - (1972)
 - Linguaggio molto "formale" (progetto accademico)
 - Programmazione *strutturata*
 - Utile per la didattica
 - Compilato



Linguaggi di programmazione

- **C**
 - Bell Labs (1972)
 - Evoluzione più efficiente del PASCAL
 - Istruzioni per ottimizzazione del codice
 - efficiente
 - Molto usato nella programmazione di sistema
 - compilato
- **C++**
 - Bell Labs ('80)
 - Evoluzione del C ad oggetti
 - Diverso paradigma di programmazione
 - Include il C
 - compilato



Linguaggi di programmazione

- Java
 - Sun Microsystems ('90)
 - Evoluzione del C++
 - Schema misto compilato+interpretato
 - Portabilità universale tramite formato intermedio (*bytecode*)
 - Vasta gamma di librerie
 - Supporto alla programmazione web
- Perl
 - GNU project ('90)
 - interpretato
 - complesso, ma molto potente
 - molto utilizzato nella programmazione web
 - programmi=>*script*