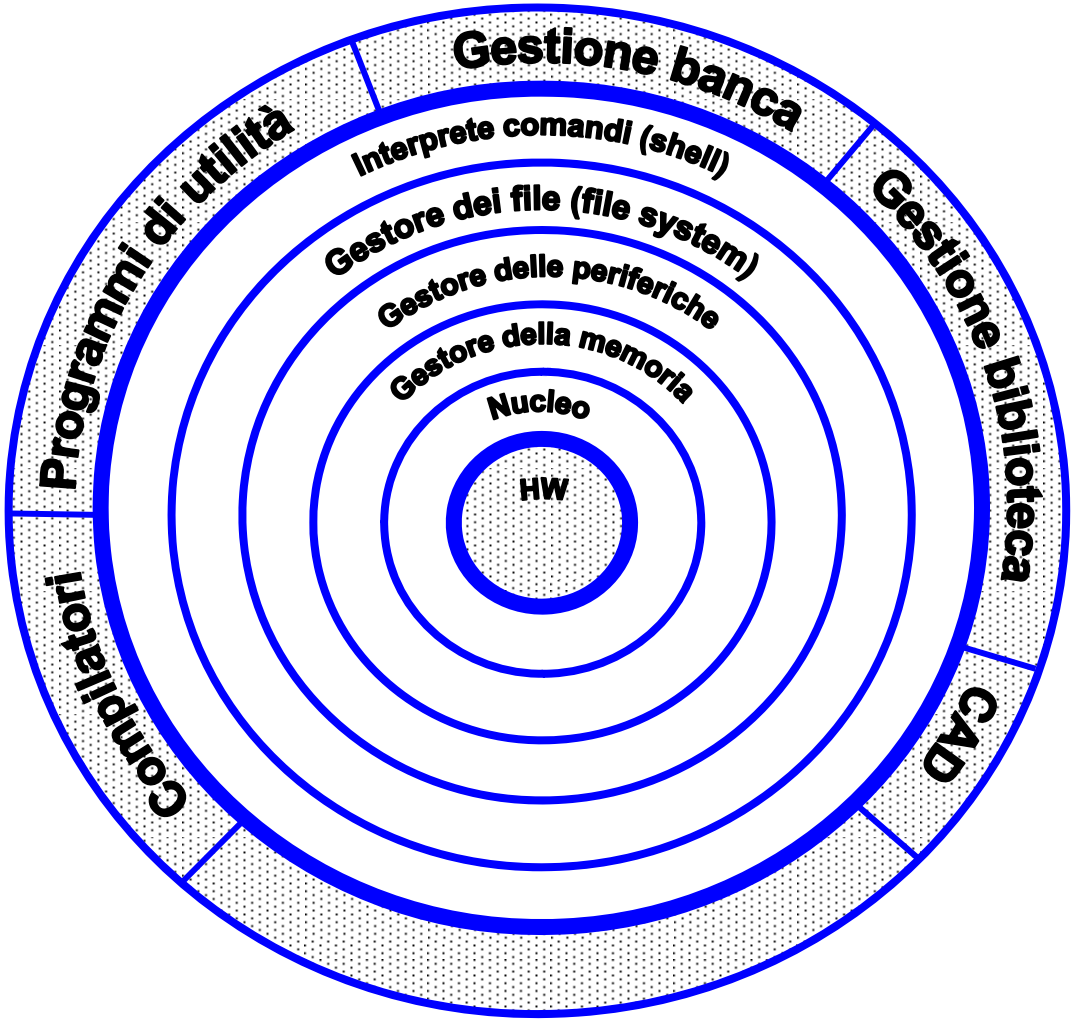
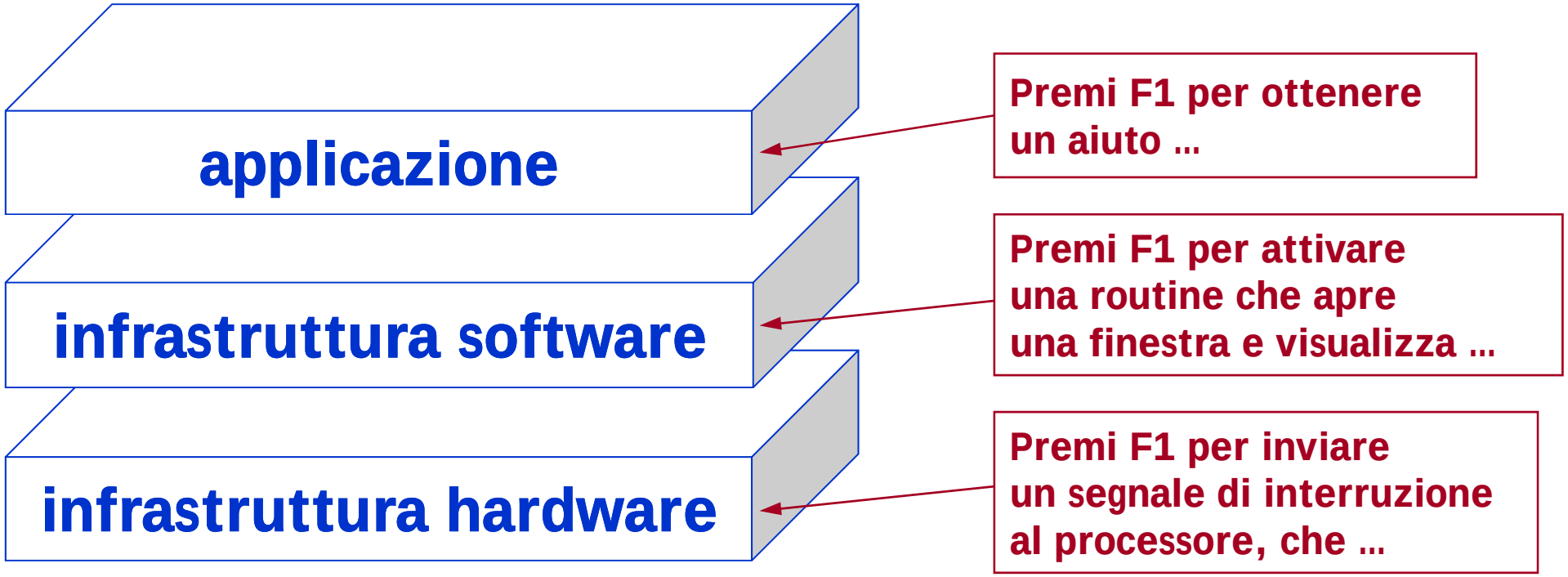
Capitolo 9
Le applicazioni

Software Applicativo



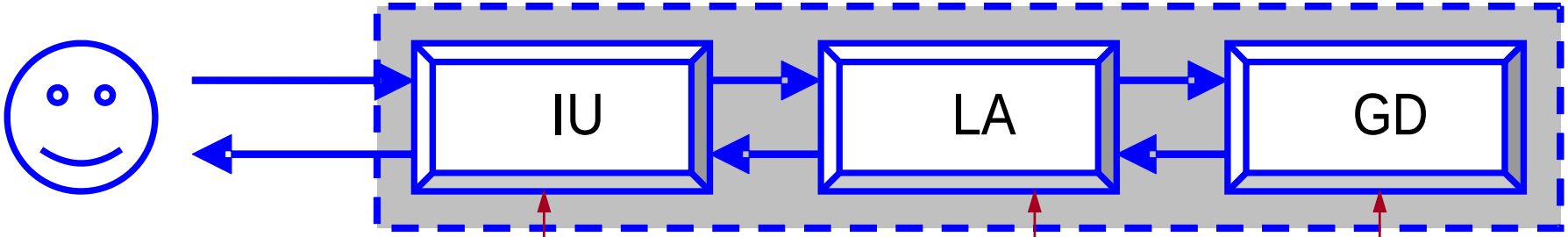
L'organizzazione "per livelli" dei sistemi informatici



... sono "punti di vista" diversi, e complementari, sullo stesso sistema...

Un'applicazione ben costruita (e funzionante...!)) può essere usata prescindendo in larga misura dalle problematiche infrastrutturali dei livelli sottostanti

L'organizzazione funzionale di un'applicazione informatica

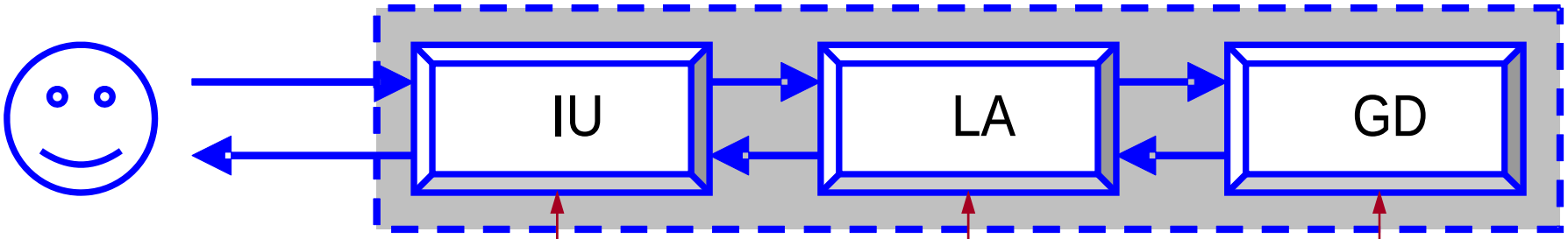


sottosistema di interfaccia con l'utente (*user interface*, o anche *presentation layer*), incaricato di controllare i comandi e i dati che l'utente fornisce come input al sistema e di presentare come output i risultati dell'esecuzione del programma

sottosistema di logica applicativa (*business logic layer*), che implementa gli specifici algoritmi di manipolazione dei dati che caratterizzano l'applicazione

sottosistema di gestione dei dati (*data layer*), che si occupa dell'organizzazione dei dati, e in particolare della loro memorizzazione e del loro reperimento efficiente

Organizzazione funzionale delle applicazioni: l'esempio dei sistemi di elaborazione di testi



interfaccia con l'utente: finestre, menu, icone, …

logica applicativa: funzioni per formattare caratteri, effettuare ricerche e sostituzioni, controllare l'ortografia del testo, …

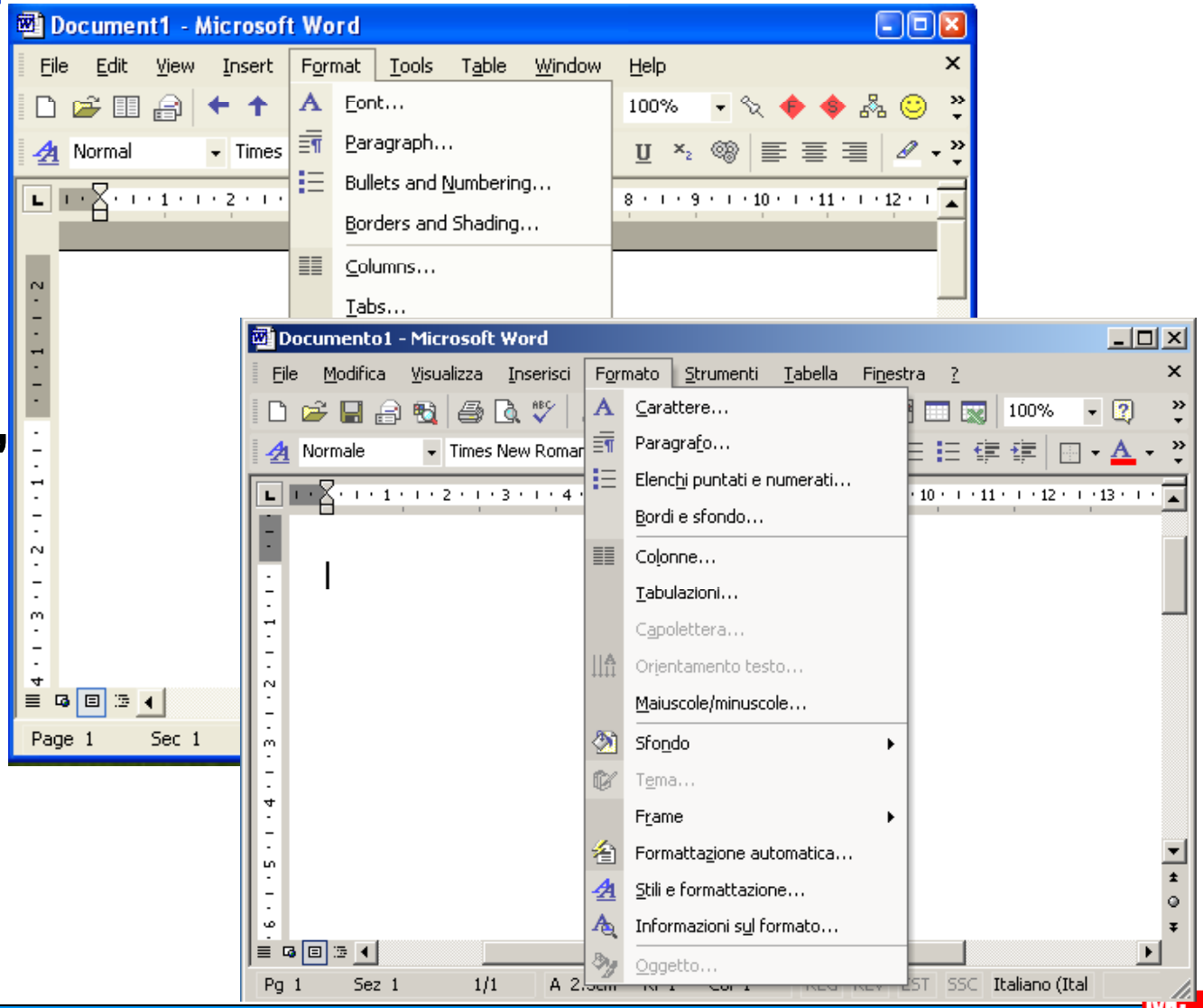
gestione dei dati: funzioni per la gestione (apertura e memorizzazione) dei file dati, dei dizionari, dei fogli di stile, …

Organizzazione funzionale delle applicazioni: indipendenza funzionale

I tre sottosistemi sono funzionalmente indipendenti l'uno dall'altro...

... si può, per esempio, cambiare IU senza modificare LA e GD

... come nel caso della localizzazione delle applicazioni...

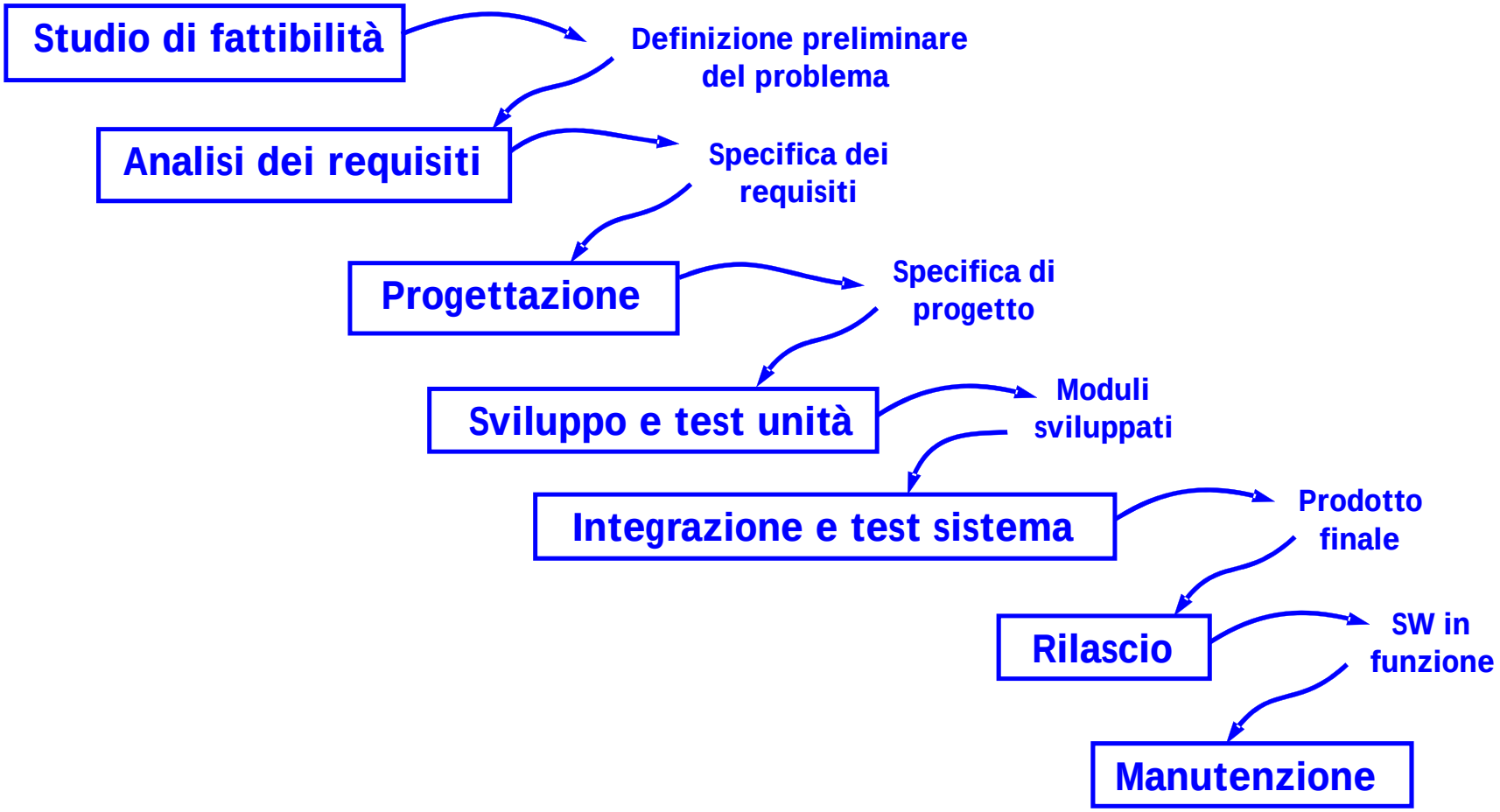


Sviluppare è più che programmare

Il processo di sviluppo delle applicazioni è tipicamente scandito nelle seguenti attività:

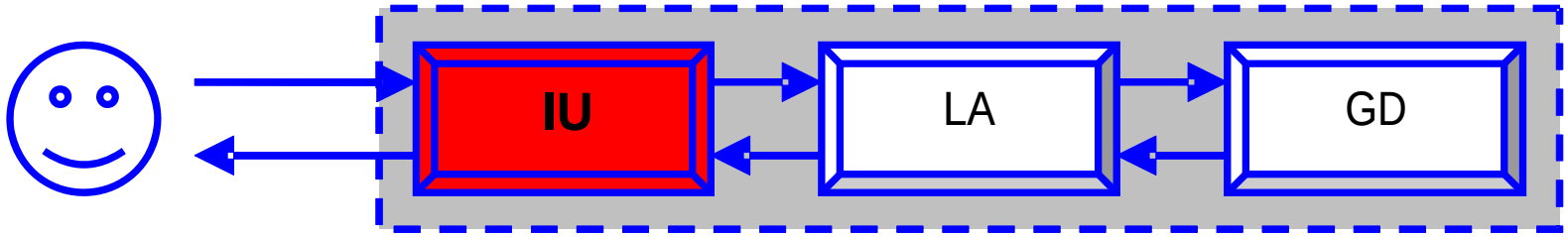
- **analisi e specifica dei requisiti**
- **progettazione**
- **sviluppo e test dell'applicazione software**
- **rilascio (deployment)**
- **manutenzione**

Il ciclo di vita di un'applicazione: il modello a cascata



Il ciclo di vita di un'applicazione: il modello evolutivo

Il sottosistema IU - interfaccia utente



- **Ha il compito di gestire l'interazione del programma con l'utente:**
  - consente all'utente di specificare gli input al programma
  - restituisce all'utente gli output del programma
  - e inoltre, fornisce informazioni sullo stato di esecuzione del programma

- **A causa dell'indipendenza funzionale tra sottosistemi, le stesse caratteristiche per LA e GD possono essere ottenute con IU diverse ...**

Il sottosistema IU – Un esempio

LA: contare il numero n di volte che un carattere x è presente in una stringa y; dunque, input=x, y; output=n
(p.es. x='e', y='E' l'esempio di un programma' ⇨ n=2)

IU (opzione 1):
interfaccia a linea di comando (*command line*), con modalità batch

```
>conta1
Conta il numero di occorrenze di un carattere all'interno di una stringa.
Se viene specificato il terzo argomento, -m, il conteggio viene effettuato
senza distinguere tra maiuscole e minuscole.

Uso: conta1 -c<carattere> -s"<stringa>" [-m]

>conta1 -ce -s"E' l'esempio di un programma"
Numero di occorrenze: 2

>_
```

Questo programma opera secondo la logica:

1. **l'utente specifica tutti i parametri di input e mette in esecuzione il programma**
2. **il programma viene eseguito**
3. **il risultato prodotto nell'esecuzione viene presentato all'utente come output del programma**
4. **l'esecuzione termina**

Il sottosistema IU – Un esempio

IU (opzione 2):
interfaccia a linea di comando (*command line*), con modalità interattiva modale

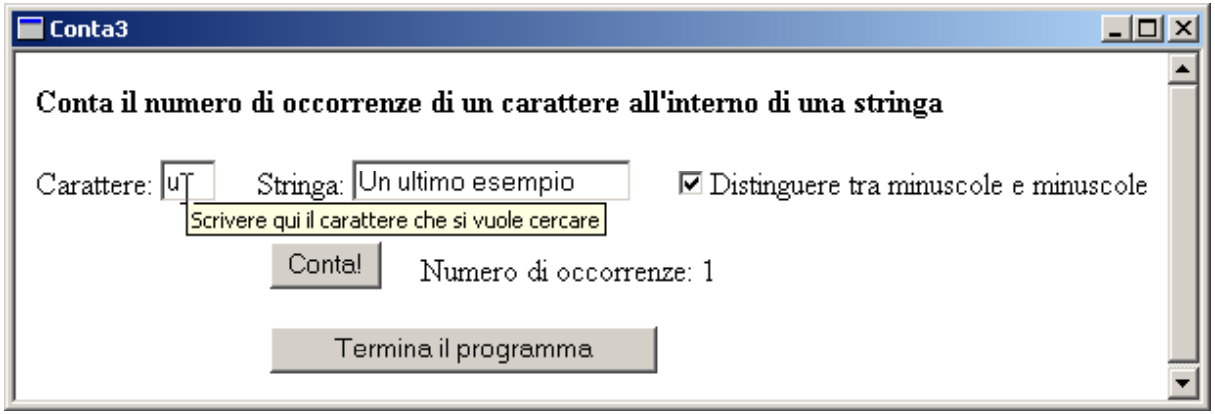
```
>conta2
Conta il numero di occorrenze di un carattere all'interno di una stringa.

Carattere: a
Stringa: Ancora una prova
Distinguere tra minuscole e minuscole (s/n)? n

Numero di occorrenze: 4

>_
```

Questo programma opera secondo la logica:

1. **l'utente mette in esecuzione il programma**
2. **il programma attiva una sessione domande-risposte per ottenere dall'utente i parametri di input**
3. **viene eseguita la parte del programma che opera sui parametri di input per ottenere il risultato**
4. **il risultato prodotto nell'esecuzione viene presentato all'utente come output del programma**
5. **l'esecuzione termina**

Il sottosistema IU – Un esempio

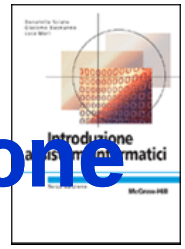
IU (opzione 3):
interfaccia grafica (*graphical user interface*), con modalità interattiva non modale



Questo programma opera secondo la logica:

1. **l'utente mette in esecuzione il programma**
2. **il programma attiva un'interfaccia utente interattiva non modale e si mette in condizione "di ascolto"**
3. **fino a che l'utente non seleziona il pulsante per terminare l'esecuzione del programma:**
   a) l'utente immette i parametri di input
   b) l'utente seleziona il pulsante che mette in esecuzione la parte del programma che opera sui parametri di input per ottenere il risultato
   c) il risultato prodotto viene presentato all'utente
4. **l'esecuzione termina**

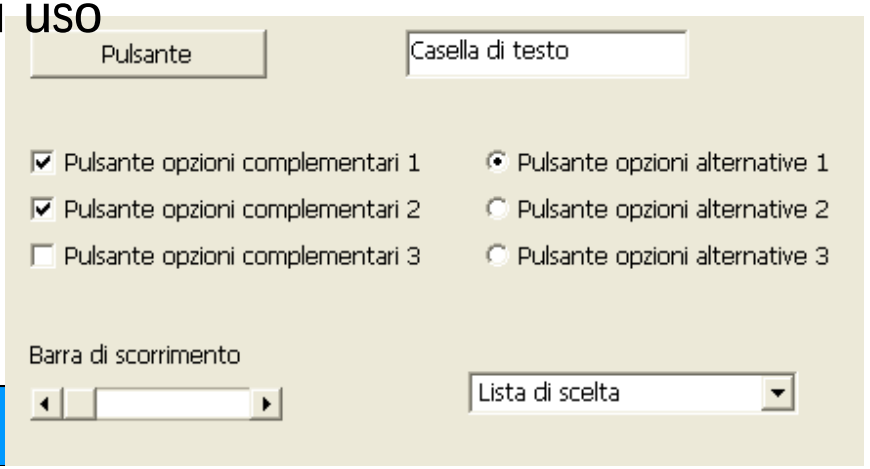
Graphical user interface (GUI): personalizzazione



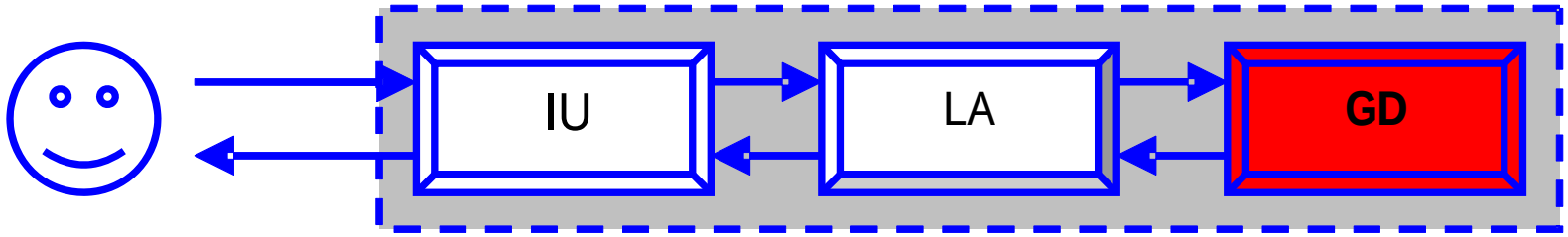
- **Forniscono una molteplicità di strumenti di input…**
  - per copiare nell'archivio appunti (clipboard) un testo selezionato si può:
    - con la tastiera premere Ctrl-C
    - con la tastiera aprire il menu Modifica con il corrispondente tasto scorciatoia e quindi muovere il cursore con i tasti freccia fino a evidenziare la voce Copia premendo quindi Invio
    - con il mouse aprire il menu Modifica e quindi selezionare la voce Copia
    - con il mouse aprire il menu contestuale (cursore sopra la selezione e click con il tasto di destra) da cui scegliere la voce Copia
    - …
- **… e consentono quindi a ogni utente di adottare un proprio stile di uso dei programmi e, spesso, di personalizzarne l'interfaccia**

Graphical user interface (GUI): standardizzazione

- **L'uniformità delle IU consente agli utenti di focalizzare l'attenzione sulle funzionalità del programma e non sull'interfaccia utente (sul cosa fare e non sul come farlo); sono stati standardizzati:**
  - l'uso di sequenze di tasti di comando, per esempio F1 per richiamare un aiuto sul programma in esecuzione o Ctrl-C per copiare nell'archivio appunti l'entità selezionata
  - la presenza e le modalità di uso dell'archivio appunti, che funziona in modo omogeneo in diverse applicazioni e consente di trasferire dati tra applicazioni
  - la presenza e le modalità di uso di funzioni di utilità di impiego generale, come Apri…, Salva con nome…, o Stampa…
  - la presentazione grafica e le modalità di uso degli oggetti grafici nelle finestre:



Il sottosistema GD – gestione dei dati

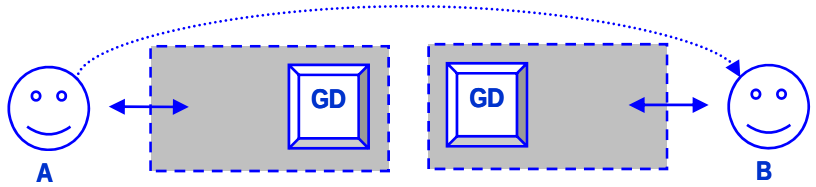


- **Ha il compito di organizzare i dati gestiti dall'applicazione, in particolare relativamente alla loro memorizzazione e al loro reperimento efficiente**
- **A causa dell'indipendenza funzionale tra sottosistemi, le stesse caratteristiche per IU e LA possono essere ottenute con GD diversi…**
- **Per esempio, una tipica applicazione di elaborazione di testi mette a disposizione diversi formati di file per memorizzare i testi**
- **Il sottosistema GD diventa critico quando utenti che usano applicazioni diverse vogliono scambiarsi dei dati**

Il sottosistema GD – Scambio di dati

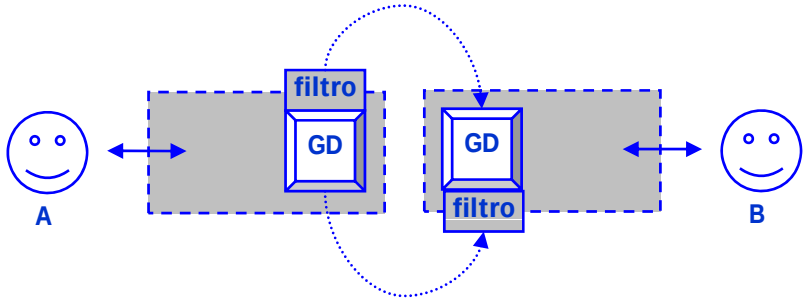
L'utente B deve operare su un file inizialmente prodotto dall'utente A
Ci sono tre possibili scenari

1.



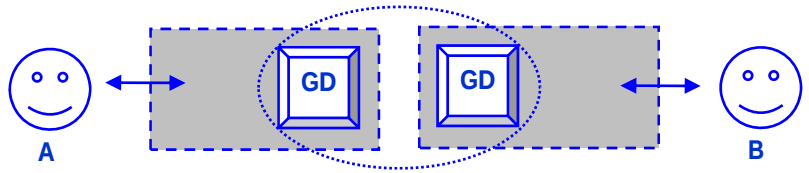
i programmi di A e B adottano formati di file non compatibili e non offrono strumenti per rendere i dati accessibili all'esterno: B deve riscrivere il testo

2.



i programmi di A e B adottano formati di file non compatibili ma dispongono di filtri di esportazione e/o di importazione: A può esportare il testo nel formato del programma di B oppure B può importare il testo dal formato del programma di A

3.



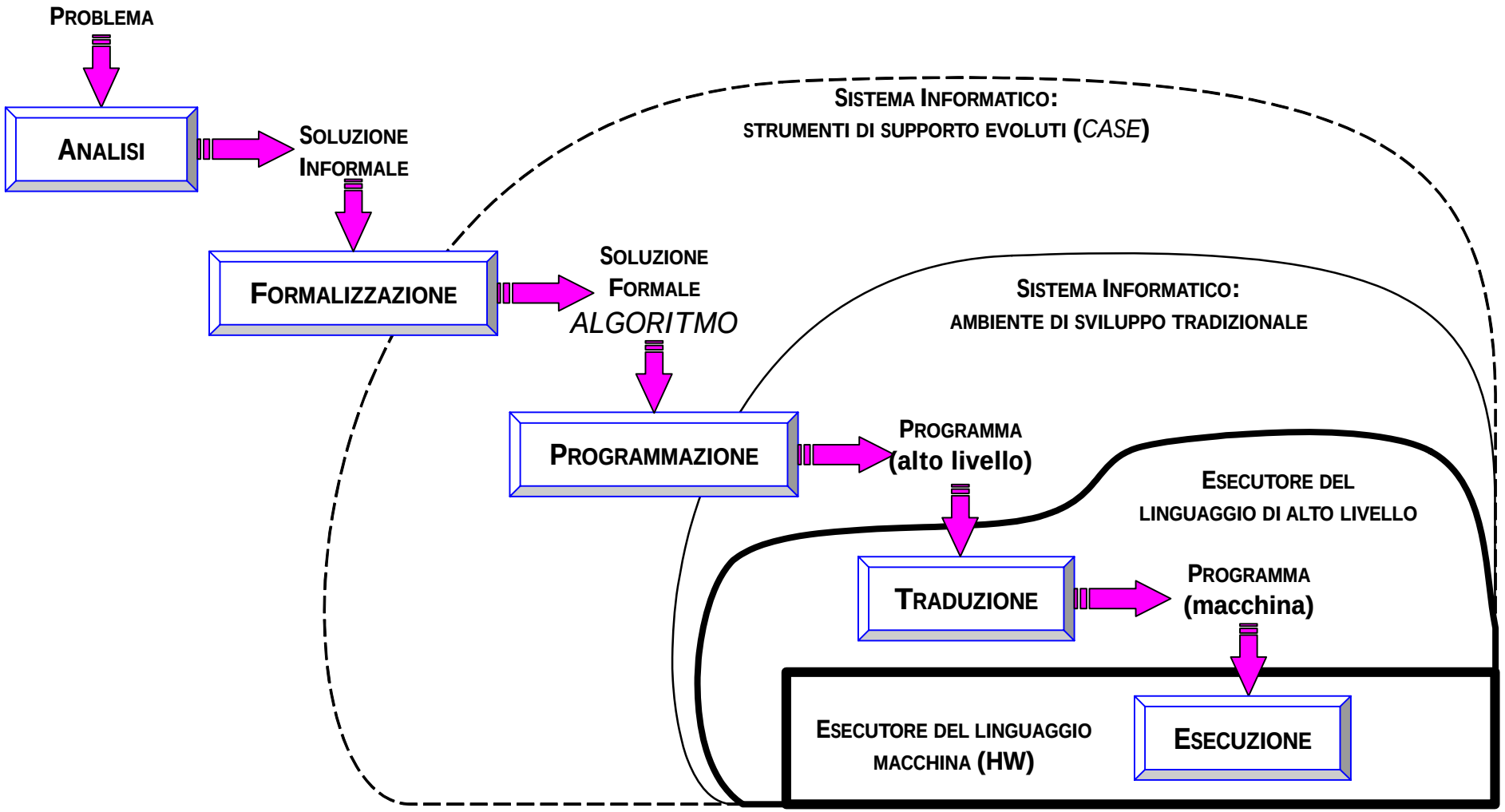
i programmi di A e B adottano uno stesso formato di file e quindi si comportano come se disponessero di uno stesso sottosistema GD: non ci sono problemi di scambio dei dati

Il sottosistema GD – Il ruolo dei DBMS

- **Una tecnologia consolidata per la gestione dei dati persistenti (cioè che devono essere mantenuti in modo permanente, e non solo durante l'esecuzione del programma): sistemi di gestione di basi di dati (*data base management system, DBMS*)**
- **I DBMS basati sul modello relazionale forniscono un'interfaccia comune per la definizione della struttura del DB, l'amministrazione del DB stesso e l'accesso ai dati, ottenuta con l'impiego del linguaggio SQL**
- **Grazie a un linguaggio standard (SQL):**
  - i dati mantenuti in un DBMS possono essere manipolati da programmi molteplici (accessibilità)
  - i dati possono essere trasferiti tra DBMS diversi e rimanere identicamente accessibili (indipendenza dal DBMS)

Sviluppo di un Programma

Sviluppo di un programma



Linguaggio di Programmazione

- La notazione con cui è possibile descrivere gli algoritmi.
- Programma: è la rappresentazione di un algoritmo in un particolare linguaggio di programmazione.
- Ogni linguaggio di programmazione dispone di un insieme di “parole chiave” (keywords)
- Ogni linguaggio è caratterizzato da una **sintassi** e da una **semantica**

Il linguaggio macchina

- **Il linguaggio macchina è direttamente eseguibile dall'elaboratore, senza nessuna traduzione.**
  - Istruzioni ed operandi relativi al programma in esecuzione sono caricati in memoria e quindi sono memorizzati in forma *binaria*.
  - Vincolo: conoscenza dei metodi di rappresentazione delle informazioni utilizzati.

- **Istruzione: carica nell'accumulatore**

10010000 11001100

Il linguaggio Assembler

- **Le istruzioni corrispondono univocamente a quelle macchina, ma vengono espresse tramite nomi simbolici (parole chiave).**
- **Il programma prima di essere eseguito deve essere tradotto in linguaggio macchina (**assemblatore**).**
- **Vincolo: necessità di conoscere in dettaglio le caratteristiche della macchina (registri, dimensione dei dati, set di istruzioni)**
- **Anche semplici algoritmi implicano la specifica di molte istruzioni**

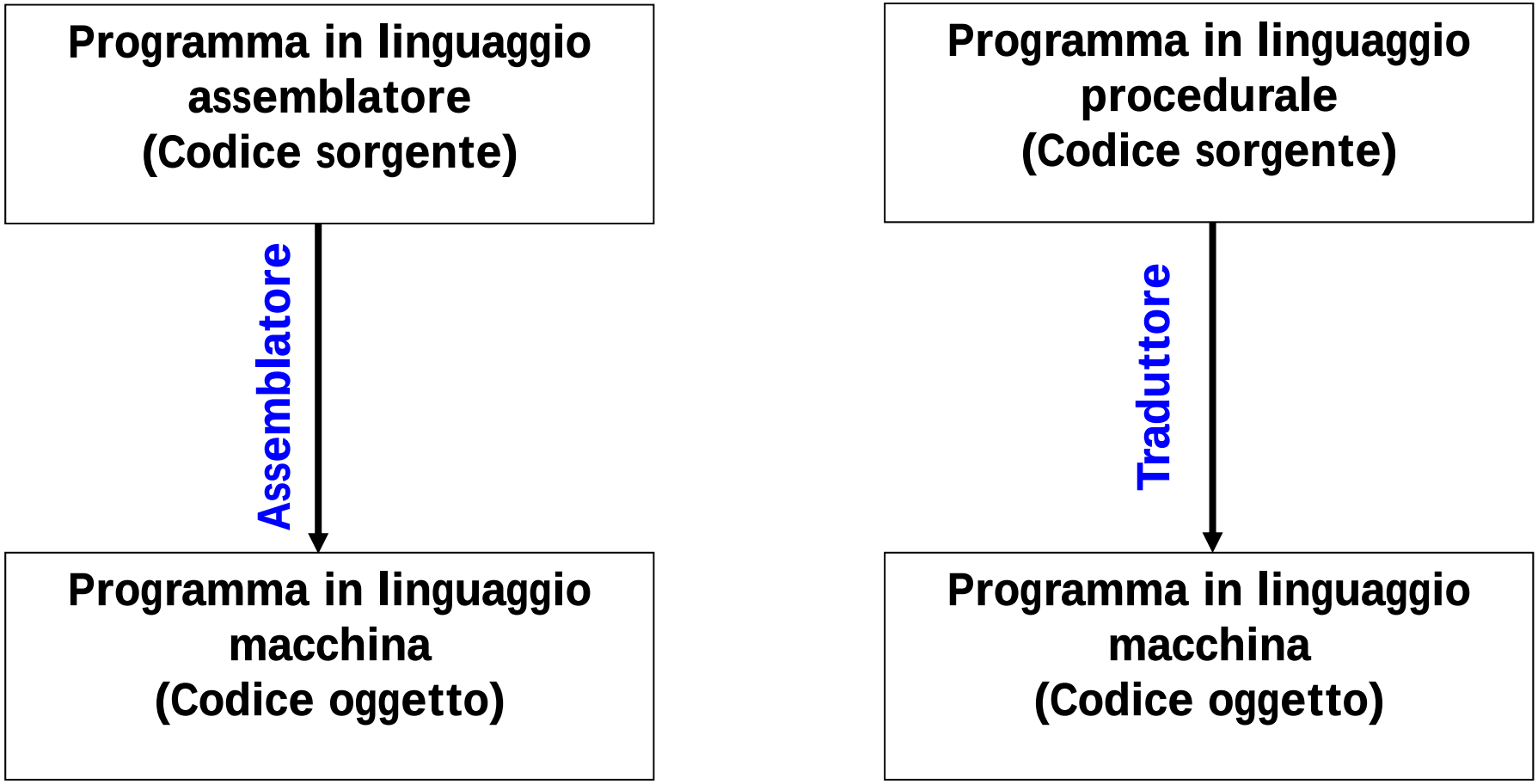
I linguaggi di alto livello

- **Sono i linguaggi di terza generazione. Le istruzioni esprimono una serie di azioni. Il programma prima di essere eseguito deve essere tradotto in linguaggio macchina (**traduttore**)**

- **Il programmatore può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico**

- **Sono indipendenti dalla macchina (astrazione)**

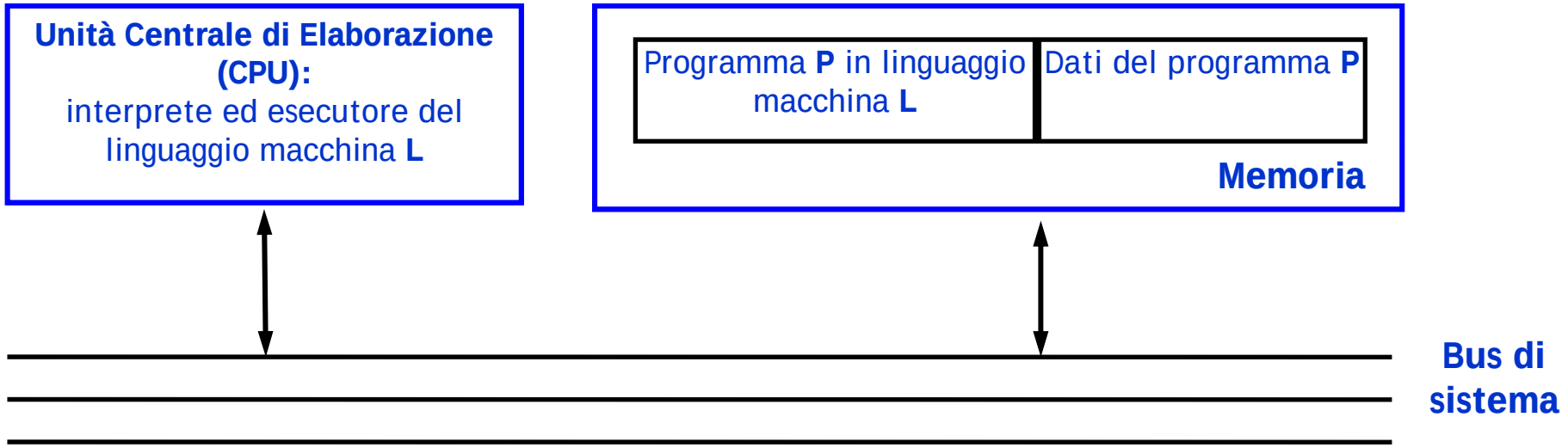
Linguaggi di II/III generazione



Sviluppo di un Programma

- I traduttori sono programmi particolari che provvedono a convertire il codice di programmi scritti in un dato linguaggio di programmazione (sorgenti), nella corrispondente rappresentazione in linguaggio macchina (eseguibili)

CPU come interprete del suo linguaggio macchina

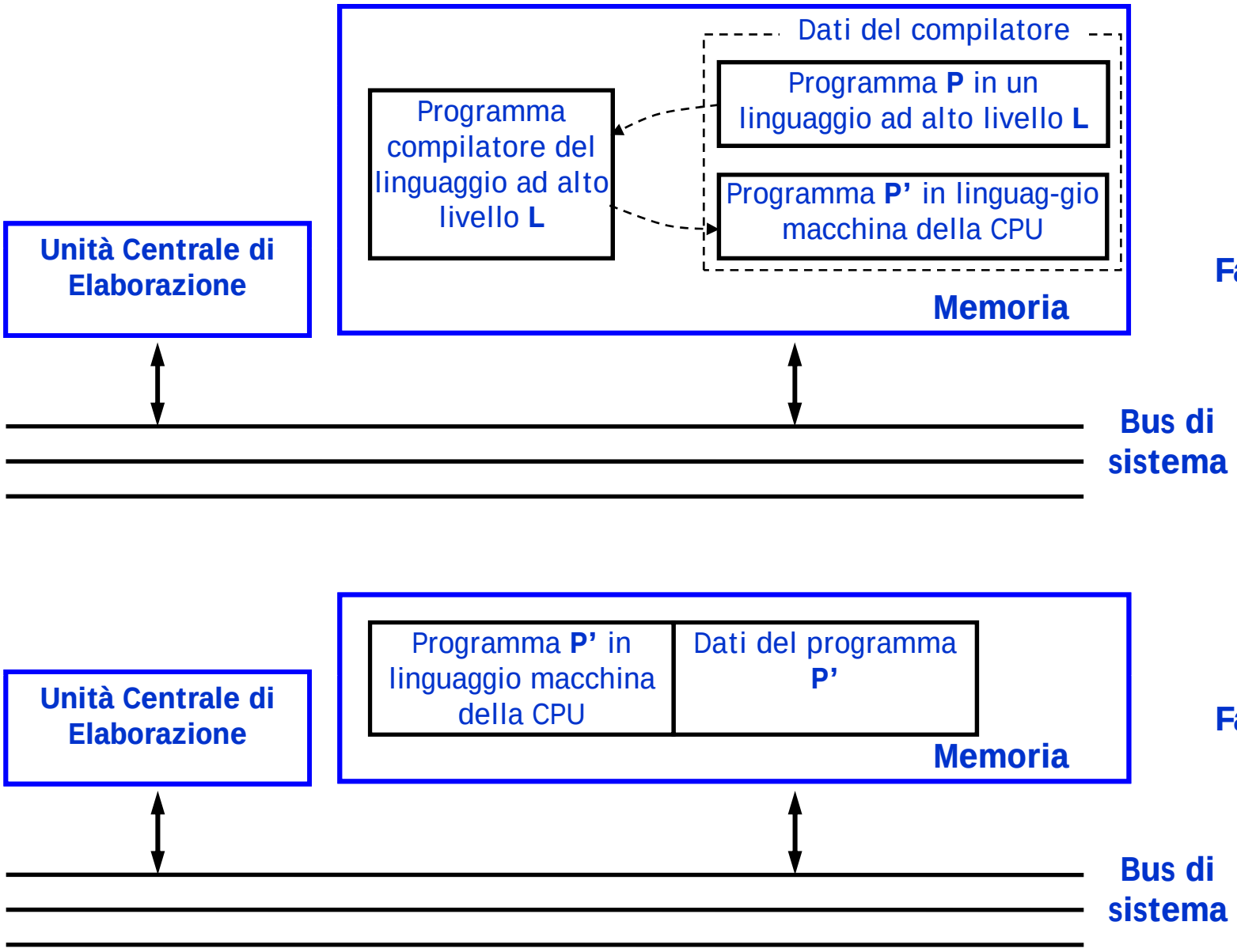


Due tipi di traduttori

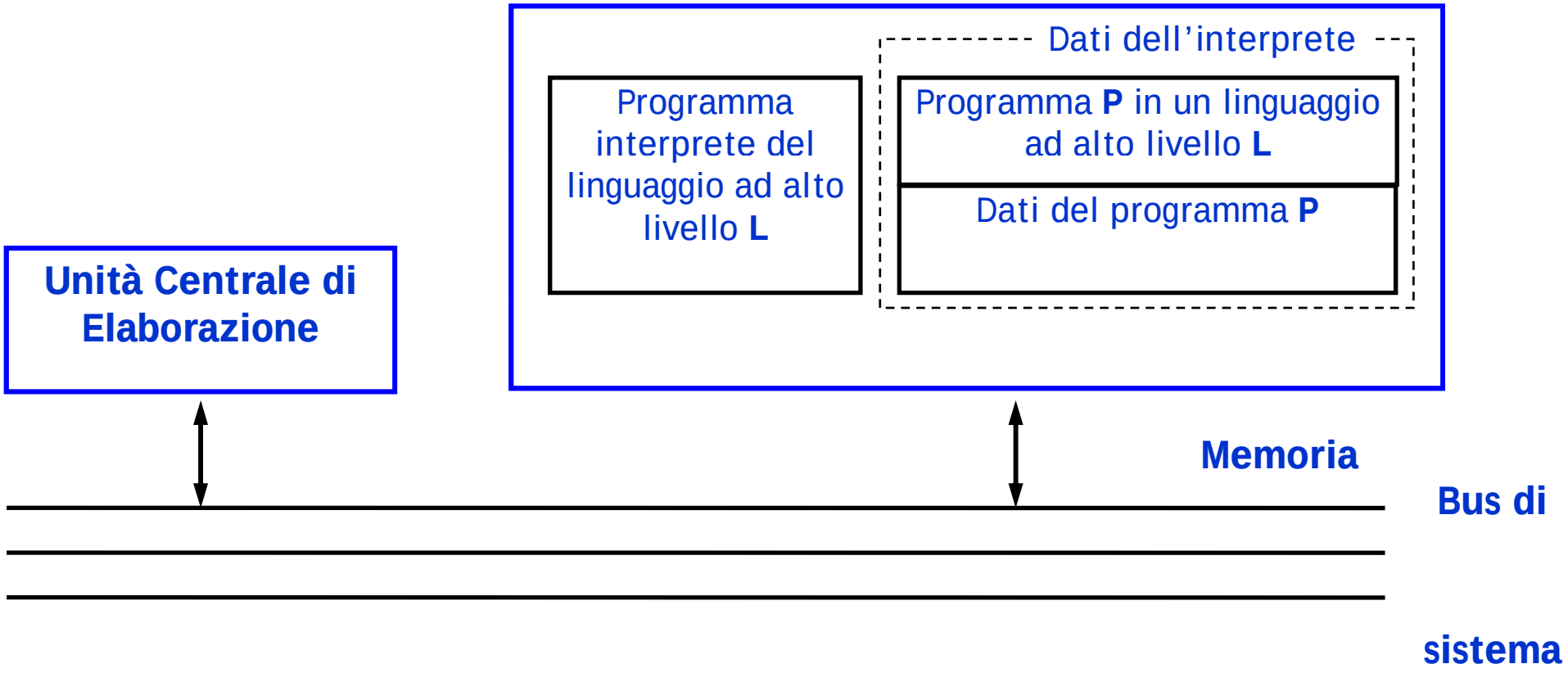
- **Compilatori**
  - Accettano in ingresso l'intero programma e producono in uscita la rappresentazione dell'intero programma in linguaggio macchina.

- **Interpreti**
  - Traducono ed eseguono direttamente ciascuna istruzione del programma sorgente, istruzione per istruzione.

Compilatore



Interprete



Due tipi di traduttori /2

- **Compilatori**
  - Per ogni programma da tradurre, lo schema viene percorso una volta sola prima dell'esecuzione.
- **Interpreti**
  - Lo schema viene attraversato tante volte quante sono le istruzioni che compongono il programma; ad ogni attivazione dell'interprete su una particolare istruzione, segue l'esecuzione dell'istruzione stessa.
- **L'esecuzione di un programma compilato è più veloce dell'esecuzione di un programma interpretato.**