17 - I linguaggi di programmazione

Sommario

- Cos'è un linguaggio di programmazione?
- Una prospettiva storica: linguaggi di prima, seconda e terza generazione
- I paradigmi di programmazione
- Principali componenti di un linguaggio di programmazione ad alto livello
- Dichiarazione di variabili e di costanti, assegnamento

Cos'è un linguaggio di programmazione? A cosa serve?

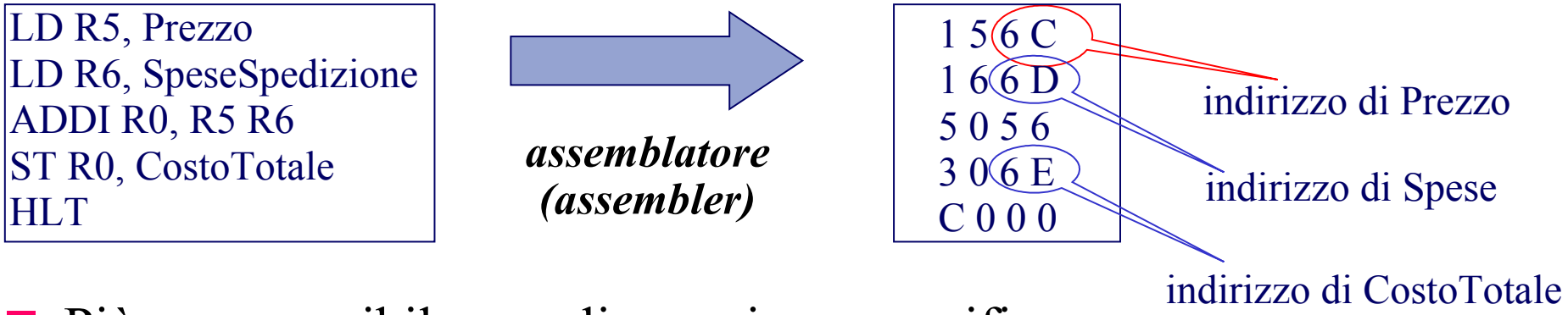
- E' un ***linguaggio formale*** definito da un insieme di regole che permettono di:
  - "generare" tutti i programmi corretti;
  - controllare se un dato programma è corretto.
- Serve per scrivere algoritmi per la soluzione di problemi in modo ***non ambiguo***.
- **Esempio:** il ***linguaggio macchina*** che abbiamo visto.
- Esistono moltissimi linguaggi di programmazione, che vengono classificati in diversi modi. Per esempio
  - ***generazioni di linguaggi***
  - ***paradigmi di programmazione***

Linguaggi della “prima generazione”: i linguaggi macchina

- Le istruzioni sono eseguibili direttamente dalla CPU
- Sono rappresentate da sequenze di bit (o cifre esadecimali)
- Fanno riferimento esplicitamente all'hardware (registri, indirizzi di celle di memoria)
- Strutture di controllo molto semplici: sequenza e salto condizionato
- I dati su cui operano sono molto semplici: numeri in complemento a due o virgola fissa, sequenze di bit
- **Svantaggi**:
  - specifico per ogni calcolatore: non è portabile (se cambia l’indirizzo di una locazione, o cambia la CPU, bisogna cambiare il programma)
  - difficile per l’umano (gestire lunghe sequenze di bit)

Linguaggi della “seconda generazione”: i linguaggi assembler

- Consentono di usare “nomi simbolici”
  - per le istruzioni *(codici operazione)*
  - per le celle di memoria *(identificatori)* invece di indirizzi
- Ogni istruzione corrisponde a un'istruzione del linguaggio macchina: l'***assemblatore*** traduce un programma assembler in linguaggio macchina, per poterlo eseguire.



- Più comprensibile per gli umani, ma specifico per ogni CPU: non portabile

Linguaggi della “terza generazione”: i linguaggi ad alto livello

- Le istruzioni primitive sono concettualmente più complesse
- Un'istruzione corrisponde a numerose istruzioni macchina/assembler
- I tipi dei dati sono più complessi e strutturati
- Le strutture di controllo sono più ricche (come nello pseudocodice)
- I programmi devono essere tradotti in linguaggio macchina per mezzo di strumenti complessi chiamati ***compilatori***.
- I programmi sono ***portabili***: possono essere compilati su macchine diverse.

```
CostoTotale = Prezzo + SpeseSpedizione
```

linguaggio di alto livello

```
LD R5, Prezzo
LD R6, SpeseSpedizione
ADDI R0, R5 R6
ST R0, CostoTotale
HLT
```

linguaggio assembler

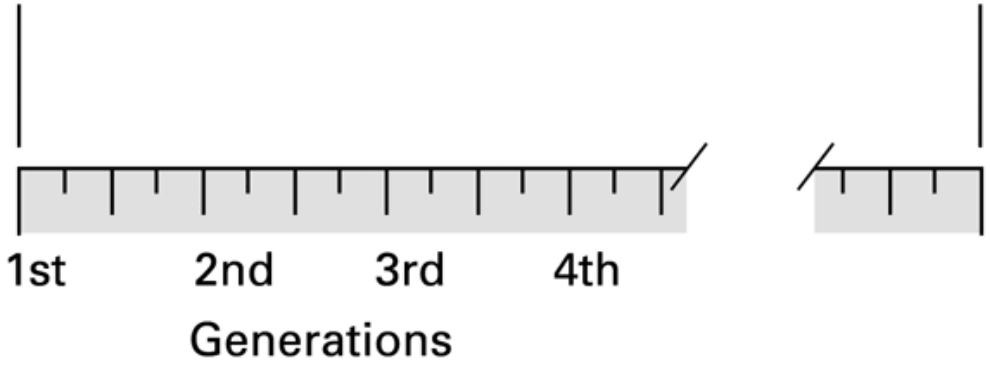
```
1 5 6 C
1 6 6 D
5 0 5 6
3 0 6 E
C 0 0 0
```

linguaggio macchina

Le generazioni dei linguaggi di programmazione

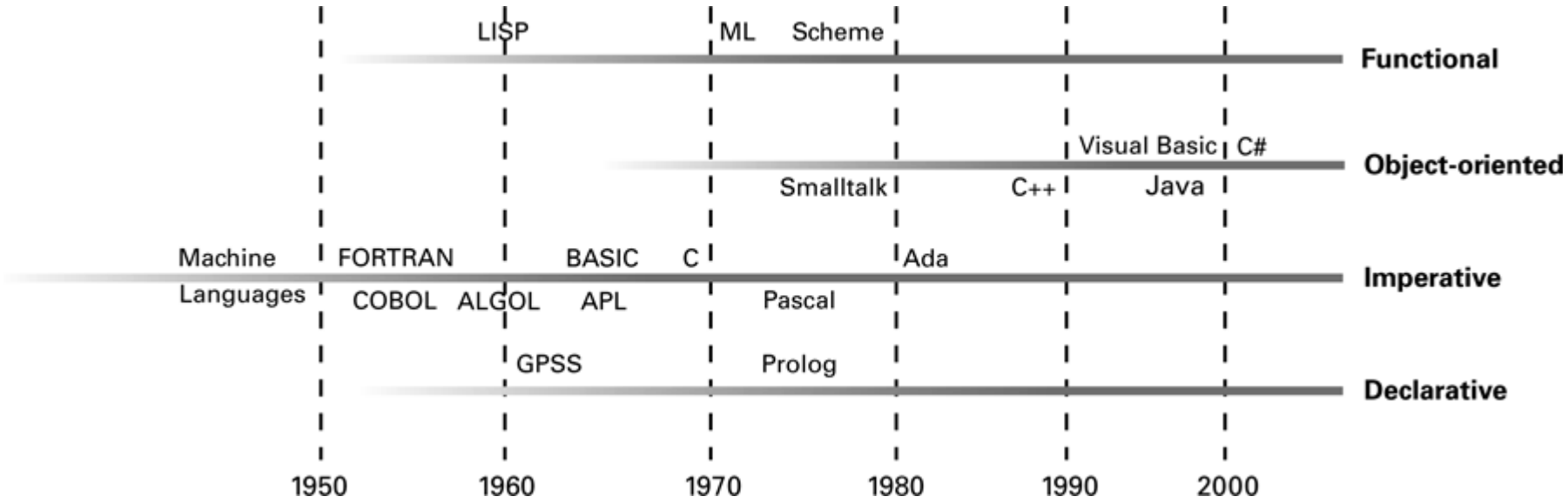
Problemi risolti in un ambiente in cui l'essere umano si adegua alle caratteristiche della macchina

Problemi risolti in un ambiente in cui la macchina si adegua alle esigenze dell'essere umano

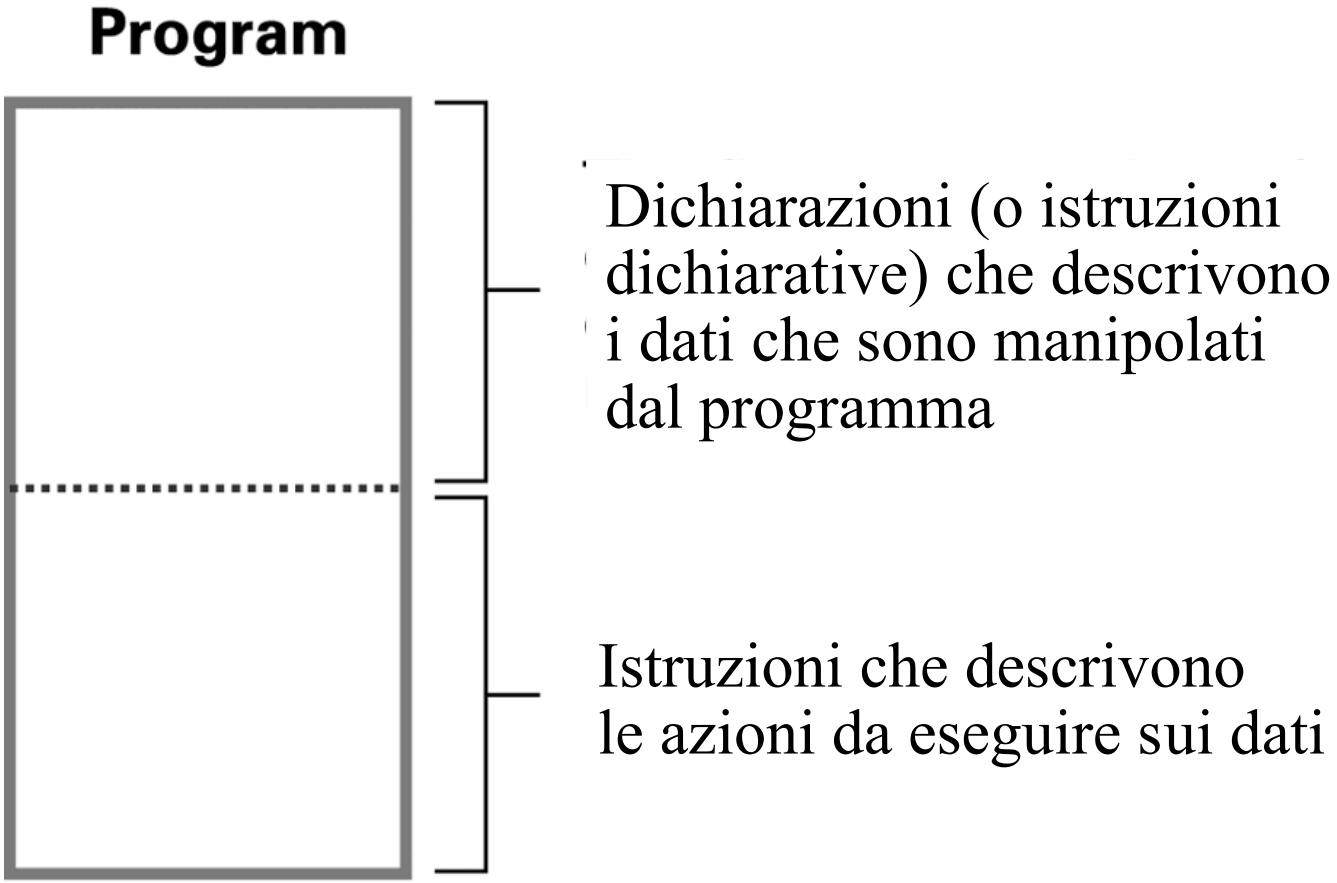


L'evoluzione dei Paradigmi di Programmazione

- **Imperativo**: algoritmo descritto come sequenza di passi
- **Funzionale**: algoritmo come composizione di ***funzioni***
- **Dichiarativo**: niente algoritmo, solo relazione tra input e output
- **Object-Oriented**: algoritmo descritto come comunità di ***oggetti*** con stato interno, che interagiscono invocando ***metodi***



Principali componenti di un programma (imperativo)



Variabili e costanti

- Le dichiarazioni introducono le **variabili**: celle di memoria caratterizzate da identificatore (il nome) e da un valore (assunto mediante *assegnamenti*)
- Ogni variabile ha un **tipo**, che determina i valori che può assumere (questo determina lo spazio necessario in memoria).
  Esempi di **tipi di dati**:
  - **booleani** (0,1),
  - **interi** (rappresentati in complemento a due)
  - **reali** (rappresentato in virgola mobile)
  - **caratteri** (rappresentati con ASCII (in C) o UNICODE (in Java))

- Si possono anche dichiarare delle **costanti:** hanno un nome e un valore come le variabili, ma il valore non può essere cambiato con un assegnamento.

Esempi di dichiarazioni di variabili e di assegnamenti (C, C++, Java...)

```
float Length, Width;  // dichiara due variabili di tipo float
int   Price, Total, Tax;
char Symbol = '*';  // dichiarazione con inizializzazione
Price = 150;          // assegnamento
Length = 74.25;
Symbol = '#';
Total = 125.76;   // NO: errore di tipo!
Width = 34;         // OK: un intero è anche un reale...
final float PI = 3.1416;  // Una dichiarazione di costante
```