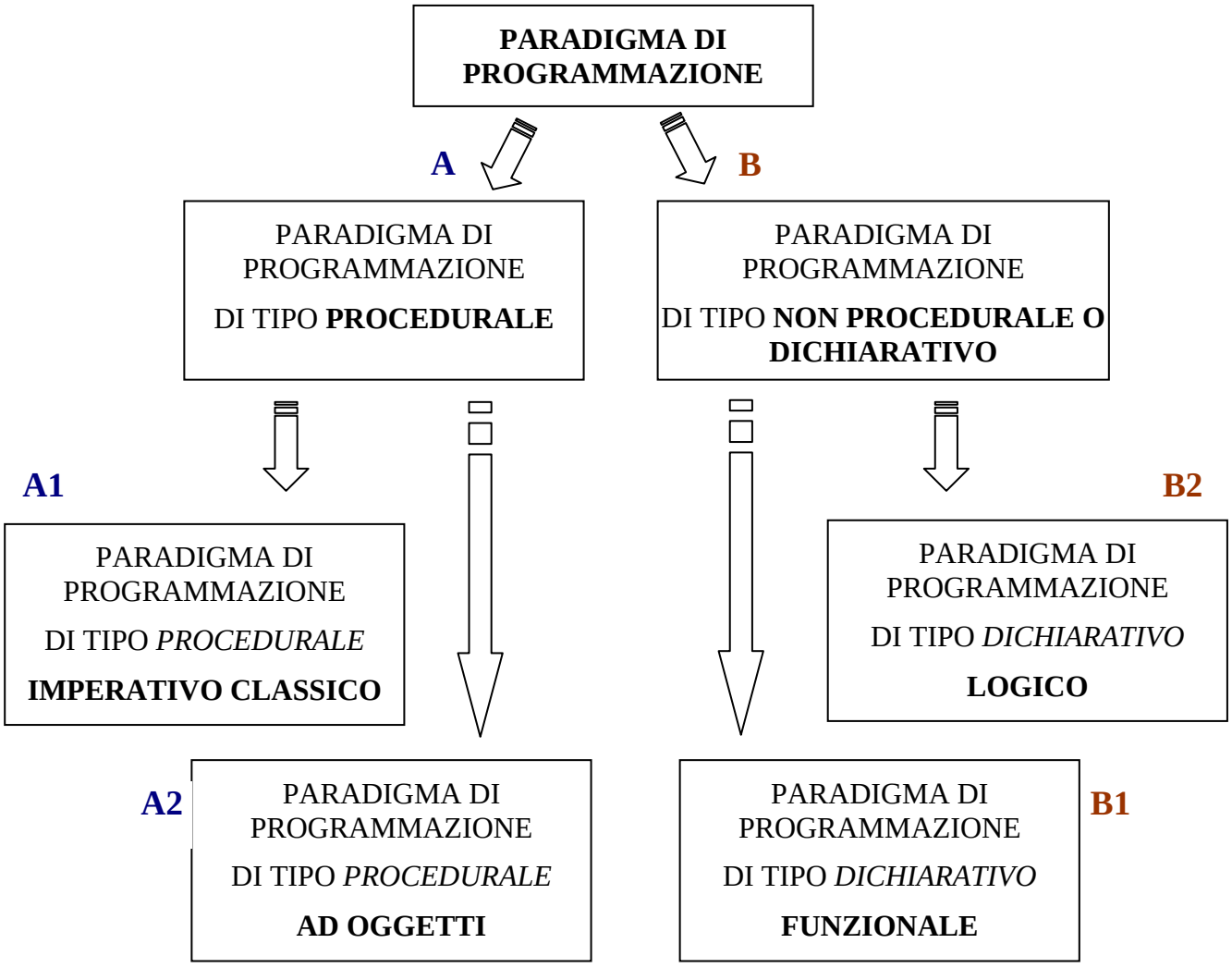
2. PARADIGMI DI PROGRAMMAZIONE

La disciplina che si occupa dell'automatizzazione dei processi risolutivi attraverso la codifica o l'implementazione prende il nome di programmazione di un algoritmo in un linguaggio di **programmazione.**
Con il termine **paradigma di programmazione** (*paràdeigma*: dal greco "modello") ci si riferisce ad un modello al quale la mente del **programmatore** deve astrattamente uniformarsi (il programmatore è colui che traduce le istruzioni dell'algoritmo descritto in pseudolinguaggio e/o con un flow-chart nel linguaggio di programmazione scelto).

Schematizzando, i paradigma di programmazione sono così suddivisibili:



A PARADIGMA DI PROGRAMMAZIONE PROCEDURALE

La programmazione basata sul paradigma procedurale si occupa del **COME** risolvere un problema.
Segue le tra fasi classiche per la risoluzione di unproblema:

- conoscenza del problema ossia studio della realtà;
- comprensione del problema;
- ricerca della soluzione che consiste nello stabilire ***come*** risolvere il problema

A1 PARADIGMA DI PROGRAMMAZIONE PROCEDURALE IMPERATIVO CLASSICO

Basato sul concetto di "*imperio*" o ***comando*** ossia su di una esplicita richiesta inviata all'esecutore.
In questo contesto il programmatore rappresenta il processo di calcolo come una sequenza di azioni.
Esempi: linguaggio **COBOL**, linguaggio **C**, linguaggio **PASCAL**, linguaggio **FORTRAN**.

A2 PARADIGMA DI PROGRAMMAZIONE PROCEDURALE AD OGGETTI (o Object-Oriented)

Basato sul concetto di "***oggetto***" che è una metafora o rappresentazione del mondo reale.
Ad ogni oggetto vengono associate informazioni e funzionalità e glòi oggetti interagiscono gli uni con gli altri.
Il programmatore che programma d oggetti deve progetare e sviluppare un insieme di oggetti software ispirati dal contesto reale del problemache si sta affrontando e che siano in grado di risolverlo.
Esempi: linguaggio **C++**, linguaggio **JAVA**

B PARADIGMA DI PROGRAMMAZIONE NON PROCEDURALE O DICHIARATIVO

La programmazione basata sul paradigma procedurale si concentra esclusivamente sul COSA richiede il problema e non si occupa minimamente sul COME ottenere i risultati.
L'idea base è quella di pervenire alla soluzione del problema dettagliando sia la conoscenza della realtà che il problema mediante opportuno linguaggio di specifica.
Il principale svantaggio di tale paradigma di programmazione è rappresentato dall'efficienza: ai risultati si perviene in tempi nettamente superiori ai casi modellati secondo il paradigma procedurale.

B1 PARADIGMA DI PROGRAMMAZIONE NON PROCEDURALE FUNZIONALE

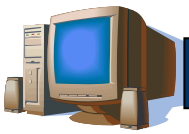
Un programma scritto in accordo a questo paradigma è costituto da un insieme di definizione di funzioni, nel senso matematico del termine. L'esecuzione del programma avviene in seguito alla richiesta fatta al sistema di valutare una espressione.
Il valore di una funzione è nota una volta noto il valore dei suoi argomenti.
La coppia (funzione, lista di argomenti) prende il nome di applicazione.
Esempi: linguaggio LISP.

B2 PARADIGMA DI PROGRAMMAZIONE NON PROCEDURALE LOGICO

Il programmatore giunge all soluzione del problema esprimendo una serie di ***assiomi***, formalizzati e comunicati al sistema, il complesso dei fatti che devono essere veri all'interno della realtà esaminata. Il risultato scaturisce dal conseguente processo di calcolo.
L'esecuzione di un programma si ottiene ponendo un ***quesito*** al sistema: la risposta consiste nella verifica di verità o di falsità dell'affermazione fatta dal quesito sulla base delle conoscenze specificate all'interno del sistema.
Esempi: linguaggio **PROLOG**.

I linguaggi di programmazione si sono evoluti di pari passo con l'evoluzione tecnologica delle macchine e per questo si parla di **generazione di linguaggi di programmazione**.

Schematizzione del processo evolutivo dei linguaggi di programmazione



BASSO LIVELLO

Facilità e velocità di programmazione

meno

più

LINGUAGGIO MACCHINA (Linguaggi di Progr. di 1° generazione)

Linguaggio influenzato completamente dall'hardware della macchina su cui doveva operare.
Per realizzare i programmi era necessario utilizzare esclusivamente i simboli binari **'0'** ed **'1'** e conoscere la CPU della macchina sulla quale si stava programmando. Tale operazione era molto complessa e venivano commessi molti errori anche per istruzioni in realtà banali.
Le operazioni possibili erano solo quelle che la macchina poteva svolgere direttamente.

LINGUAGGI ASSEMBLY (Linguaggi di Progr. di 2° generazione)

Linguaggi detti "machine oriented" (orientati alla macchina).
Tali linguaggi introdussero per la prima volta la presenza di un programma traduttore (**programma assembler** o **assemblatore**) in codice binario delle istruzioni da eseguire, legato all'hardware della macchina su cui si operava. In questo caso a disposizione del programmatore venivano messe per la prima volta dei "comandi" più vicini al suo modo di pensare (ADD, JMP, etc.) che il programma assemblatore si preoccupava di tradurre in codice binario comprensibile dalla CPU.

LINGUAGGI AD ALTO LIVELLO (Linguaggi di Progr. di 3° generazione)

Secondo questi linguaggi un programma è costituito da un insieme di istruzioni slegate dal funzionamento fisico della macchina capaci però di modificarne il contenuto della memoria.
Sono linguaggi "problem-oriented" ossia orientati al problema e vengono suddivisi a seconda del modello di programmazione (paradigma) utilizzato. (Qui ci sono quasi tutti i linguaggi di programmazione più noti)

LINGUAGGI AD ALTISSIMO LIVELLO (Linguaggi di Progr. di 4° generazione)

Sono i linguaggi **UHLL (ultra high level language)** ed hanno la caratteristica di consentire all'utente di risolvere problemi utilizzando un linguaggio simile a quello utilizzato nella propria attività. (Es. fogli elettronici e linguaggi per la manipolazione delle basi dati)

LINGUAGGI AD ALTISSIMO LIVELLO (Linguaggi di Progr. di 5° generazione)

Si stanno studiando e stanno cominciando ad essere utilizzati nel campo dell'applicazione dell'intelligenza artificiale che si propone la risoluzione di problemi ritenuti di competenza dell'intelletto umano **(sistemi esperti)**

più

meno

Efficienza del programma

ALTO LIVELLO