- **INFORMATICA**

⇒ che cosa si intende per informatica?

▪ Per informatica si intende la scienza che si occupa della rappresentazione e dell'elaborazione dell'**informazione**:
la quale rende possibile gestire e organizzare i dati.

⇒ L'Informatica studia:

▪ i **procedimenti algoritmici** per risolvere problemi;
▪ i **linguaggi** di **programmazione** per descrivere algoritmi;
▪ le architetture dei **computer** per eseguire programmi;
▪ il ragionamento **automatico**.

- **CONCETTO DI ASTRAZIONE**

⇒ L'informatica si basa su concetti propri, come quello di **ASTRAZIONE**;

di cui l'obiettivo è quello di creare il **giusto modello** per un problema e individuare le **tecniche appropriate** per risolverlo in maniera automatica, tramite un **calcolatore**.

- **PROCEDIMENTO RISOLUTIVO**

⇒ Per rendere il procedimento eseguibile in maniera automatica:

▪ deve essere una **sequenza finita** di operazioni **DEFINITE**;
▪ deve essere specificato l'**ordine di esecuzione** di ogni operazione;
▪ deve essere specificata la **condizione di termine** della sequenza;
▪ deve, a parità di dati iniziali, portare sempre alla **soluzione** (che deve essere sempre la stessa)

- **ALGORITMO** (procedimento astratto)

⇒ Per far sì che il procedimento risolutivo per risolvere un problema si automatizzi necessario un **ALGORITMO**:

cioè un procedimento sistematico, costituito da una **sequenza finita di operazioni definite**, ognuna delle quali è precisa ed eseguibile, da applicare ai dati in **input** perché possa fornire dei dati in **output**.

In qualsiasi caso il procedimento avrà termine e fornirà una risposta precisa in un **numero finito di passi**. (concetto di FINITEZZA)

Una volta definito, l'algoritmo deve essere sottoposto ad un **esecutore**.

- **L'ESECUTORE**

⇒ L'algoritmo deve essere **sottoposto ad un esecutore**, il quale deve essere in grado di:

▪ **interpretare** la sequenza di comandi; (senza ambiguità)
▪ **eseguire** i comandi forniti;
▪ **memorizzare** informazioni che permettano di accedere ad esse e modificarle.

L'esecutore **non deve necessariamente essere consapevole di quello che sta facendo** perchè il numero di passi è finito e deve solo attenersi ad eseguire il procedimento.

Dunque la **descrizione di un algoritmo è indipendente dall'esecutore** che dovrà eseguirlo.

- **PROGRAMMA**

⇒ La **rappresentazione dell'algoritmo** comprensibile ed eseguibile dall'esecutore automatico costituisce un **PROGRAMMA**, la cui elaborazione delle azioni è richiesta tramite **comandi elementari** chiamati **ISTRUZIONI** espresse attraverso il **LINGUAGGIO di PROGRAMMAZIONE**.

Dunque un **PROGRAMMA** è la **formulazione testuale**, in un certo linguaggio di programmazione, di un **algoritmo** che risolve un dato problema.

- **LINGUAGGI DI PROGRAMMAZIONE**

⇒ Un algoritmo può essere implementato in **linguaggi diversi** attraverso vari tipi di linguaggi di programmazione;

tuttavia ognuno dei programmi ottenuti è **equivalente** agli altri essendo un'implementazione dell'algoritmo originale.

⇒ L'uso di un linguaggio di programmazione permette di realizzare un programma che:

- implementa l'algoritmo in **maniera precisa**, in un linguaggio ad **"alto livello"**;
- non dipende dal **calcolatore** su cui viene eseguito.

- **DIAGRAMMI DI FLUSSO (FLOW CHART)**

⇒ I Diagrammi di flusso, o **flow chart**, definiscono e rappresentano in maniera non ambigua le **istruzioni** che costituiscono l'algoritmo.

Esso consente la **modellazione grafica** di un algoritmo ed è **IMMEDIATO**;
consente di descrivere un algoritmo concentrandosi principalmente sulla **sequenza delle operazioni** di cui si compone.

⇒ Si basa su alcuni **blocchi** che si differenziano per la loro forma associata alla funzione che svolgono.

- **PARALLELOGRAMMA:** blocco di input/output
- **RETTANGOLO**: blocco di sequenza che applica un' **assegnazione/operazione** e spesso implica l'uso delle **variabili**
- **ROMBO**: blocco di selezione (viene usato per controllare una **condizione** che può essere verificata o non verificata)
- **OVALE**: blocco di inizio/ fine programma

- **VARIABILE**

⇒ Le variabili sono una **porzione di memoria** che contengono dati utili e che possono essere modificati durante l'esecuzione del programma. (tramite operazioni)

Esse possiedono un **identificativo**, un **tipo** e un **valore**.

- **COSTANTE**

⇒ E' un oggetto il cui valore rimane **invariato** durante l'esecuzione di un algoritmo.

- **ISTRUZIONE DI ASSEGNAZIONE**

⇒ L'istruzione di assegnazione permette di attribuire un **valore** ad una **variabile**.

E' importante notare che l'assegnazione è:

- **conservativa a destra**: la variabile a destra dell'assegnazione non cambia valore tra prima e dopo l'esecuzione, mantiene il suo **RV**;
- **distruttiva a sinistra**: la variabile a sinistra dell'assegnazione perde il valore che aveva prima dell'assegnazione;

Ad esempio, per **y=x**, l'assegnazione pone il **RV(x)** nel contenitore di posizione **LV(y)**.

- **FLUSSO DI ESECUZIONE**

⇒ Quando definiamo un algoritmo, possiamo avere un'estrema variabilità in quello che è l'**ordine di esecuzione** della sequenza di istruzioni che compongono l'algoritmo.

Il **Flusso di esecuzione** serve a specificare se, quando, in quale ordine e quante volte devono essere eseguite le istruzioni dell'algoritmo.

Questo viene controllato dai **COSTRUTTI DI PROGRAMMAZIONE** inseriti in un programma.

- **COSTRUTTI DI PROGRAMMAZIONE**

⇒ I costrutti di programmazione sono strumenti per costruire **istruzioni composte** a partire dalle istruzioni semplici.

Esistono due tipi di costrutti di controllo:

- Costrutti **selettivi** : (**if-then** e **if-then-else**)
  i quali consentono di operare delle **scelte** in funzione della **condizione** inserita nel rombo.
- Costrutti **iterativi**: (**for**, il **while**, **do-while** e il **repeat-until**)
  sono detti **cicli**, consentono di ripetere alcune operazioni un certo numero di volte.
  Hanno un' **inizializzazione**, una **condizione**, e una **modifica**.

PATTERN

⇒ I design pattern sono degli **schemi algoritmici generali**, un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la **progettazione** e lo **sviluppo del software**.

I **vantaggi** nell'utilizzo dei pattern sono i seguenti:

- i **tempi di sviluppo** saranno inferiori;
- la **qualità** sarà maggiore perché usando *pattern noti* si ha garanzia di **funzionamento** e assenza di **errori** (bug)
- si arriva alla **soluzione** più efficiente tra quelle disponibili
- è molto più **pratico** per apportare modifiche

- **INFORMAZIONE**

⇒ Per Informazione si intende la **raccolta/conoscenza** di dati che permettono di ridurre il grado di incertezza in merito ad una particolare situazione.

Quindi per scambiare un'informazione si presuppone l'esistenza di due entità:
un **mittente**, il quale fornisce l'informazione, e un **ricevente**, il quale la riceve.

Il concetto di informazione è legato al **concetto di scelta**; quanto più ampia è la scelta maggiore è l'informazione che si riceve.

In un algoritmo, l'informazione assume tre caratteristiche fondamentali:

- **ATTRIBUTO,** che indica il **contesto** in cui si trova una determinata informazione
- **TIPO,** è l'**insieme (finito)** di tutti i possibili valori su cui operare la scelta.
- **VALORE,** è la **scelta** di uno specifico elemento all'interno del **Tipo**.

(Si ottiene, dunque, un'informazione quando un attributo assume un valore di un determinato tipo.)

- **RAPPRESENTAZIONE DI UN INFORMAZIONE**

⇒ Esistono **infiniti modi** per rappresentare un'informazione.

Tra tutti i metodi adottabili, quello più importante nei sistemi informatici è la **rappresentazione BINARIA** (o **codifica**):

essa si basa sulla rappresentazione nelle due cifre binarie (**0** e **1**), dette **digit**;

(da cui **rappresentazione digitale**.)

> (**Definizione**: Sia data un'informazione di tipo **T** di cardinalità **n**, e un alfabeto di **k** simboli **A** = {s1, s2, ..., sk}; sia inoltre **S** l'insieme di tutte le stringhe (o configurazioni) composte da **m** simboli di **A**.)

- **LINGUAGGI DI PROGRAMMAZIONE**

⇒ Anche le **istruzioni**, così come le informazioni, si possono **codificare** come sequenze di **bit**.
Nello specificare un'istruzione bisogna precisare l'operazione da compiere e i dati coinvolti in essa.

Essendo l'insieme delle diverse operazioni un **insieme finito** è possibile codificarlo con un **certo numero** di **bit** (**codice operativo**) e quindi un esecutore permette l'esecuzione di un programma, cioè di una **sequenza di istruzioni** che realizzano un particolare algoritmo e che sono descritte nel linguaggio interpretabile dal calcolatore.

I **vantaggi** per l'uso di linguaggi ad "alto livello" sono:

- realizzare un programma che implementa l'algoritmo in maniera precisa, utilizzando una **qualità** adeguata che rende i programmi facilmente **manutenibili** e **riusabili**.
- **trascurare i dettagli** relativi alla rappresentazione dei **dati** all'interno dei registri
- realizzare un programma che **non dipende** dal calcolatore sul quale è stato realizzato

⇒ I linguaggi di programmazione ad alto livello sono linguaggi in cui la **SINTASSI** e la **SEMANTICA** sono definite tramite regole **rigide** e **precise**, così da **eliminare** le ridondanze tipiche del linguaggio comune e così da **tradurre** (tramire i compilatori) un programma scritto in un *linguaggio ad alto livello* ad uno in *linguaggio macchina.*

- **SISTEMA DEI TIPI**

⇒ Per ogni informazione usata in un programma è necessario specificare il **tipo** entro cui essa appartiene.
A questo scopo, ogni linguaggio di programmazione mette a disposizione del programmatore un insieme di **tipi predefiniti** detto **sistema dei tipi**.

- tipi **NUMERICI**:
  - **intero** (costituito da un sottoinsieme **limitato** dei numeri interi)
  - **reale** (costituito da un sottoinsieme **limitato** e **discreto** dei numeri reali)
  - **reale doppia precisione**
    (costituito da un sottoinsieme **limitato** e **discreto** dei numeri reali, ma con range e precisione maggiore)

- tipi **NON NUMERICI**:
  - **carattere**
  - **logico** (È un tipo costituito dai due soli valori: **vero** e **falso**)
  - **stringa** (Consente di trattare un'informazione costituita da una **sequenza di caratteri**)

- **SISTEMA DEGLI OPERATORI**

⇒ **OPERAZIONI LOGICHE**

- L'operazione **A and B** restituisce **true** se e solo se sia **A** che **B** sono **true**, altrimenti restituisce **false**
- L'operazione **A or B** restituisce **false** se e solo se sia **A** che **B** sono **false**, altrimenti restituisce **true**
- L'operazione **not A** restituisce **true** se **A** è **false**, restituisce **false** se **A** è **true**

⇒ **PRECEDENZA**

- Gli operatori unari **+** e **–** e quello logico **not** hanno precedenza maggiore degli altri operatori aritmetici, relazionali e logici.
- Gli operatori **aritmetici** hanno priorità maggiore rispetto agli operatori logici e di quelli relazionali.
- Tra gli operatori **aritmetici**, ***** e **/** hanno la medesima priorità, maggiore della priorità di **+** e **-**, che hanno la stessa priorità.
- l'operatore **and** ha priorità maggiore di quella di **or**.

⇒ **ASSOCIATIVITA'**

- L'associatività di un operatore è una proprietà che determina come vengono raggruppati gli operatori della **stessa precedenza** in assenza di parentesi
- Un operatore è associativo a **sinistra** se, **a parità di priorità**, viene applicato da **sinistra** verso **destra**.