INFORMATICA

scienza che studia la RAPPRESENTAZIONE e ELABORAZIONE dell' INFORMAZIONE

ES: 83640 ← è un DATO dotato di SIGNIFICATO

↓

è solo un numero naturale, NON È UN'INFORMAZIONE perché privo di significato.

DATO → VALORE, TIPO

TIPO → insieme di possibili valori

1) INFORMAZIONE → deriva da "informare", "dare forma" ed è qualcosa che viene comunicato in una qualsiasi forma scritta o orale.

2) ELABORAZIONE → ha il compito di trasformare un'informazione per ottenere un certo risultato.

- → viene fatto da un ELABORATORE/CALCOLATORE (Pc) che assume il compito di gestire e memorizzare le informazioni e i dati.
- → bisogna dare delle istruzioni alla macchina attraverso l'ALGORITMO →

Una sequenza FINITA di PASSI (o azioni elaborative) necessari per risolvere AUTOMATICAMENTE un problema o portare a termine un compito.

↳ ES: Le ricette culinarie sono un esempio di SEQUENZA FINITA.

ESECUTORE

- COMPRENSIONE: è in grado di leggere e capire
- ISTRUZIONE: è in grado di compiere i singoli passi.

↳ * LINGUAGGIO DI PROGRAMMAZIONE: 1) C (linguaggio alto livello) 2) C++ (plus plus)

LINGUAGGIO DI ASTRAZIONE → Grazie a ciò si può effettuare un'astrazione POSITIVA eliminando il SUPERFLUO.

↓

Questi linguaggi necessitano di una TRADUZIONE che viene effettuata attraverso altri macchinari

↳ Le ISTRUZIONI vengono date dall' ALGEBRA DELLA LOGICA DELLE COMUNICAZIONI.

LINGUAGGIO MACCHINA O ASSEMBLATIVO → È il linguaggio del computer che non capisce il linguaggio C o C++

* è un linguaggio che ha una sintassi sua → NON AMBIGUITÀ.

3) RAPPRESENTAZIONE → permette di rappresentare un'informazione
→ è FINITA
→ implica l'esistenza di un CODICE → insieme di regole e convenzioni da seguire.

ALGORITMO

Si compone di 2 SEQUENZE

STATICA

Sequenza fissa di passi elaborativi

DINAMICA

Sequenza in cui l'elaborazione potrebbe non coincidere con quella Statica.
Ciò dipende dai dati che stiamo elaborando.

Prima di effettuare un programma bisogna partire da una FORMA A PRIORI → INIZIALIZZAZIONE

ES: Sommare due numeri di tre cifre

$$\begin{array}{r} 353\ + \\ 918\ = \\ \hline 1271 \end{array}$$

SEQUENZA STATICA

1) Scrivere i numeri che vogliamo sommare, incolonnati
2) Consideriamo le due cifre più a destra ed a un riporto uguale a zero
3) Calcoliamo somma e riporto delle tre cifre appena considerate
4) Scriviamo la somma calcolata opportunamente incolonnata

PASSO DECISIONALE - SEQUENZA DINAMICA

5) Se ci sono cifre a sinistra di quelle appena considerate, torno al passo 3
6) Se l'ultimo riporto è diverso da zero, scrivilo nel risultato

ES: In questo caso la sequenza statica coinciderà con la sequenza dinamica

SEQUENZA STATICA = SEQUENZA DINAMICA

$$6+1=7\times 6=42:2=21$$

1) Scegliere un numero da 1-9
2) Addizionarlo con 1
3) Moltiplicarlo per il numero iniziale
4) Dividerlo per 2

} PASSI

Abbiamo così ottenuto la somma dei primi n numeri interi, in questo caso $n=6$

$$6+5+4+3+2+1=21$$

$$\frac{n\times(n+1)}{2}=21$$

ESPRESSIONE

- **NUMERICA** → Può assumere come valori i numeri
- **LOGICA** → Può assumere solo 2 VALORI → VERO / FALSO

OPERATORI RELAZIONARI

Sono in grado di mettere in relazione 2 quantità.

$>$ $<$ $\geq$ $\leq$ → Devono essere definiti da una ↳ **RELAZIONE D'ORDINE** → è una relazione che viene definita con i numeri reali / caratteri dell'alfabeto

$=$ $\neq$ → Sono validi per tutti

PROPOSIZIONI

- **COMPLESSE** → Sono unite dai **CONNETTIVI LOGICI**: AND, OR, NOT
- **SEMPLICI**

per rappresentarle si usano le LETTERE MAIUSCOLE

TENDENZA ALLA SEMPLICITÀ → bisogna assumerla come una sorta di PRINCIPIO.

A ⟵ CONDIZIONI SEMPLICI ⟶ B

Queste possono essere espresse solo attraverso l'uso di un operatore razionale, quindi non possono essere espresse per es: "A + B"

PRINCIPIO DI DUALITÀ → È presente nell'algebra Booleana e sostiene che se una proprietà è VERA, è vera anche la ↓

PROPRIETÀ DUALE → si ottiene sostituendo AND con OR, OR con AND

COSTANTI LOGICHE ← [VERO con FALSO, FALSO con VERO]

LA PRECEDENZA DEGLI OPERATORI → Così come nelle operazioni matematiche si dà la precedenza a moltiplicazioni e divisioni e poi a addizioni e sottrazioni, qui:

1) NOT
2) AND
3) OR

ES: Quindi "A OR B AND NOT C" è come se fosse:

"A OR [B AND (NOT C)]"

↳ in realtà le parentesi non ci vogliono

TABELLA DI VERITÀ → È una tabella che dà tutti i possibili valori che un'espressione può assumere (dati i valori degli operandi).

ALGEBRA BOOLEANA (Boole)

VERO ← LOGICA A DUE VALORI → FALSO

INSIEMI — CIRCUITI — PROPOSIZIONI

→ sono ISOMORFE tra loro, cioè se una di queste è vera, allora tale proposizione sarà vera anche nelle altre proprietà.

3 OPERATORI O CONNETTIVI LOGICI

AND — OR — NOT

AND, OR: OPERATORI BINARI O DUALI

NOT: OPERATORE UNARIO

AND → È UN OPERATORE DI DISGIUNZIONE (·)

→ È UNA PROPOSIZIONE VERA SE E SOLO SE ENTRAMBE SONO VERE

A	B	A AND B
F	F	F
F	V	F
V	F	F
V	V	V

→ 6 variabili perché:
- 2 valori (A e B) } 2×2=4
- 2 valori (V e F) }
- 2 variabili (V e F) } 2 — 6

OR → OPERATORE DI CONGIUNZIONE (+)

→ È UNA PROPOSIZIONE FALSA SE E SOLO SE ENTRAMBE SONO FALSE

A	B	A OR B
F	F	F
F	V	V
V	F	V
V	V	V

SEQUENZA FISSA →

NOT → QUI ABBIAMO SOLO DUE VARIABILI (¬)

A	NOT A
F	V
V	F

In base a questi 3 operatori/connettivi logici costruiremo poi le 14 PROPRIETÀ DELL'ALGEBRA BOOLEANA →

LE 14 PROPRIETÀ DELL'ALGEBRA BOOLEANA (7×2)

1) COMMUTATIVA
- A OR B = B OR A
- A AND B = B AND A

→ tramite il principio di dualità

2) ASSOCIATIVA
- A OR (B OR C) = (A OR B) OR C
- A AND (B AND C) = (A AND B) AND C

→ qui ci sono le parentesi perché solo "or" o solo "and"

3) IDEM POTENZA
- A OR A = A
- A AND A = A

ES: Tabella di verità →

A	B	A OR B
F	F	F
F	V	V
V	F	V
V	V	V

4) ASSORBIMENTO
- A OR (A AND B) = A
- A AND (A OR B) = A

→ viene chiamata così perché B viene assorbito da A.

↳ Per verificarla bisogna utilizzare le tabelle di verità.

Quando ci sono due espressioni che hanno tra loro un diverso grado di complessità (una più semplice e un'altra più complessa) allora nel primo caso di OR, se una delle due è VERA allora la proposizione è SICURAMENTE VERA.

ES: Se A è vera, allora A AND B è sicuramente VERA.
Se A è falsa, allora A AND B è sicuramente FALSA.

5) DISTRIBUTIVA
- A OR (B AND C) = (A OR B) AND A (OR C)
- A AND (B OR C) = (A AND B) OR A (AND C)

↓

In questo caso le parentesi non sono necessarie, si possono anche evitare

6) MINIMO E MASSIMO
- A OR V = V
- A AND F = F

↓ è il FALSO ↓ è il VERO

7) COMPLEMENTO
- A OR NOT A = V
- A AND NOT A = F

→ Grazie a queste si possono effettuare le SEMPLIFICAZIONI

TEOREMA DI DE MORGAN

1) NOT (A OR B) = NOT A AND NOT B
2) NOT (A AND B) = NOT A OR NOT B*

→ qui sono necessarie le parentesi

DIMOSTRAZIONE: attraverso le tabelle di verità.

Dimostrata una l'altra discende PER DUALITÀ.

Avendo 2 espressioni più o meno complicate il nostro obiettivo è capire se sono EQUIVALENTI, MANIPOLANDOLE vediamo se con una siamo in grado di ottenere l'altra.

ESEMPIO DELL'ADDETTO ALLE POSTE

Consideriamo 2 formulazioni APPARENTEMENTE DIVERSE e proviamo a verificare che queste due siano EQUIVALENTI (ciò lo si può verificare tramite le tabelle di verità).

2 FORMULAZIONI COMPOSTE
- 1) L'addetto accetta il numero di bollettini già pagati se MINORE DI 5 OPPURE se non ci sono altri utenti in coda.
- 2) L'addetto accetta il numero di bollettini se NON ACCADE che il numero di bollettini pagati è ALMENO 5 e ci sono altri utenti in coda

A ≡ num_bol < 5 (coincide) B ≡ ci sono utenti in coda A' ≡ num_bol ≥ 5

[A e A' sono l'uno il negato dell'altro]

A' ≡ not A

1) A OR NOT B

2) NOT (A' AND B) → deriva da NOT (NOT A AND B)

SI PUÒ RISOLVERE IN 2 MODI
- tramite le tabelle di verità per verificarne la validità
- APPLICANDO IL TEOREMA DI DE MORGAN*

ossia applichiamo una MANIPOLAZIONE sulla seconda per ottenere la prima.

↳ NOT A' OR NOT B = NOT (NOT A) OR NOT B = A OR NOT B

↳ è EQUIVALENTE alla prima!

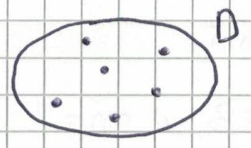
RAPPRESENTAZIONE

COMPUTER

- Attraverso il LINGUAGGIO DI PROGRAMMAZIONE avremo un'interazione con lui;
- Ha CAPACITÀ DI MEMORIZZAZIONE FINITA
- 2 PROBLEMI PRINCIPALI
  - come rappresentare nel pc un risultato/numero infinito
  - può capitare che un programma come risultato alla somma di due numeri positivi, dia un numero negativo → IMPOSSIBILE!

CODIFICA → viene considerato un sinonimo di RAPPRESENTAZIONE ma in realtà la RAPPRESENTAZIONE È UN TIPO PARTICOLARE DI CODIFICA.

→ CODIFICARE UN INSIEME (D) DI ELEMENTI



Obiettivo: giungere alla rappresentazione degli elementi utilizzando un altro ALFABETO

FUNZIONE DI CODIFICA → $R = \{a, b, c\}$ → codominio

ALFABETO CODICE

STRINGHE → sequenze di elementi che appartengono a R di LUNGHEZZA pari a m

Il CODOMINIO è costituito da elementi di R^m

PRODOTTO CARTESIANO di R con se stesso per m volte

Prodotto tra 2 insiemi è dato dall'insieme di tutte le coppie che si possono ottenere considerando come primo elemento della coppia un elemento del primo insieme e come secondo elemento della coppia, un elemento del secondo insieme.

$$\underbrace{R \times R \times \ldots \times R}_{m} \qquad R \times R = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$$

PER EFFETTUARE UNA CODIFICA → è un insieme di elementi che appartengono a R^m ossia un insieme di stringhe di LUNGHEZZA m derivanti da tutte le possibili combinazioni dei SIMBOLI che fanno parte di un ALFABETO CODICE.

Le FUNZIONI CODICE che andremo a definire avranno come CODOMINIO: R^m

funzione ← $C: D \to R^m$

↳ dominio

Una CODIFICA è → una funzione C che fa corrispondere elementi del DOMINIO D in elementi di R^m (stringhe di lunghezza m).

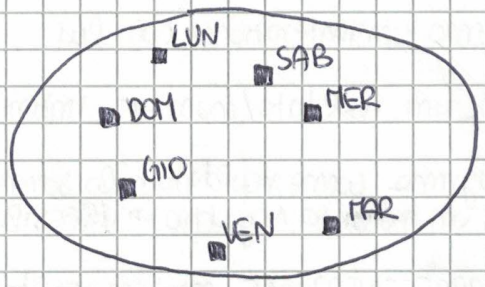
DIFFERENZA

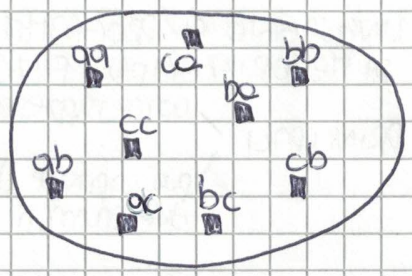
CODIFICA	RAPPRESENTAZIONE
è una generica funzione	è una "codifica scelta"

$C: D \to R^m$

D → GIORNI DELLA SETTIMANA

$R^m = \{aa, ab, ba, bb, bc, cb, cc, ca, ac\}$





$|D| = 7$ (elementi) quindi $R^m \geq 7$ perché altrimenti non avremmo un corrispettivo rappresentante per ogni elemento di D.

↳ cardinalità

Se $|R| = 3$ e $m = 2$ allora $|R^m| = |R|^m = 9$

quindi è verificata la condizione di codifica → $|R^m| \geq |D|$

Se consideriamo $|R| = h$ e nota la cardinalità di D ($|D|$) come facciamo a stabilire il valore di m (lunghezza della stringa)?

funzione

$$m = \left\lceil \log_{|R|} |D| \right\rceil$$

→ questa funzione prende il nome di CEILING

CEILING DI UN ARGOMENTO → è il primo intero maggiore uguale dell'argomento, quindi si arrotonda per eccesso.

ES: $\left\lceil \log_3 7 \right\rceil$ → $3^1 = 3 < 7$ NO
$3^2 = 9 > 7$ SI!

↳ Quindi 2 è la dimensione della stringa considerata
↳ (solo numeri $\in \mathbb{N}$)

CODICE → definizione BIUNIVOCA

CODICE INCOMPLETO → un codice si dice incompleto se il numero del valore codice è strettamente maggiore dell'insieme di valori che vogliamo rappresentare.

$|R^m| \geq |D|$

↳ quando ci sono alcune parole codice che non vengono utilizzate (es: 2)

CODICE RIDONTANTE → un codice si dice ridontante quando la lunghezza della parola codice è strettamente maggiore del valore m minimo necessario

↳ qui le parole codice sono più lunghe della dimensione minima necessaria.

↳ di conseguenza per definizione un codice ridontante è SEMPRE INCOMPLETO

ma al contrario se un codice è incompleto ciò non vuol dire che sia necessariamente ridontante.

ESEMPIO DELLE PROVINCE

↳ Sono $107 = |D|$, ci serve una sigla per rappresentarle

ALFABETO CODICE → $|R| = 26$ (lettere dell'alfabeto inglese in MAIUSCOLO)

$m \rightarrow m = \lceil \log_{26} 107 \rceil \rightarrow 26^1 = 26$ NO < 107
$26^2 = 676$ SI! > 107

↳ CEILING $= 2$ che forma infatti le sigle, es: NA, AV, ...

STRINGHE di due lettere MAIUSCOLE dell'alfabeto italiano

Per evitare di avere un codice ridondante UTILIZZIAMO 2 LETTERE E NON 3

Il codice risulta essere INCOMPLETO perché ci sono alcune coppie di lettere inutilizzate (es: ZZ

ALFABETO CODICE DEL PC → deve essere DEFINITO e UGUALE PER TUTTI

è stato scelto quindi il CODICE BINARIO costituito da 2 SIMBOLI

0 1

CIFRE DELL'ALFABETO BINARIO → BIT

Binary digIT (dig. binaria)

Quindi in questo caso $|R| = 2$ $R = \{0, 1\}$

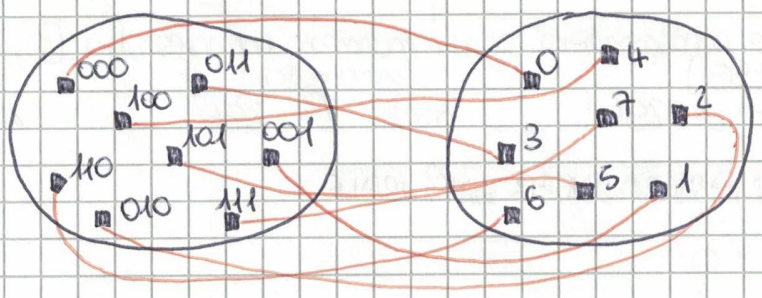
↳ stringhe 2^m avremo sempre logaritmi in base 2

CONCETTO DI FINITEZZA → coincide con l'INTERVALLO DI VALORI che si possono rappresentare

ES: se si hanno m BIT allora l'intervallo sarà: $[0 \ldots 2^m - 1]$

CON 3 BIT POSSIAMO AVERE 8 COMBINAZIONI → $2^3 = 8$

R^m (stringhe di bit) D (numeri naturali)



TABELLINA

000	→	0
001	→	1
010	→	2
011	→	3
100	→	4
101	→	5
110	→	6
111	→	7

VALORE MASSIMO

$m = 32$ quindi $2^{32} = 4$ miliardi → un numero più grande di questo non potrà essere rappresentato nei programmi

↓

LIMITE DELL'ELABORATORE

RAPPRESENTAZIONE DEI NUMERI NATURALI (interi senza segno)

BASE 10 (ALFABETO DECIMALE)
↳ CARDINALITÀ DELL'INSIEME

ES: 357 è un numero naturale composto da 3 CIFRE che sono coefficienti di determinate POTENZE di base 10.

↳ RAPPRESENTAZIONE POSIZIONALE → ciascuna cifra ha un valore che dipende dalla posizione che assume

$$\ulcorner 3\,5\,7 \urcorner$$

$$3 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0$$

$$300 + 50 + 7 = 357$$

RAPPRESENTAZIONE DEI NUMERI BINARI

BASE 2 (ALFABETO BINARIO)
↳ CARDINALITÀ DELL'INSIEME

coefficiente di potenze di base 2

ES:

$$\ulcorner 1\,1\,0 \urcorner$$

$$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

$$4 + 2 + 0 = 6$$

(cifra più significativa) msb ⌐ ⌐ lsb (cifra meno significativa)

$$\ulcorner 1\,1\,1_2 \urcorner$$

$$1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$4 + 2 + 1 = 7_{10}$$

Stringa di 8BIT → $\boxed{10110010}$

può essere sia un numero dell'alfabeto binario che un numero reale
per capire che tipo di numero è si usa il PENDICE PER INDICARE LA BASE.

10110010_{10} (numero naturale)

10110010_2 (codice binario)

in questo caso tale codice corrisponderà ad un numero compreso tra [0 ... 255]
perché $2^{(8)} = 256$ (numero di bit)

COME PASSARE DA UN CODICE BINARIO AD UN NUMERO NATURALE

$$1\,0\,1\,1\,0\,0\,1\,0$$

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 178_{10}$$

$$\times 1 \quad \times 0 \quad \times 1 \quad \times 1 \quad \times 0 \quad \times 0 \quad \times 1 \quad \times 0$$

Ciò vuol dire che se scriviamo 178 sul computer, esso dopo la CODIFICA ottiene 10110010_2.