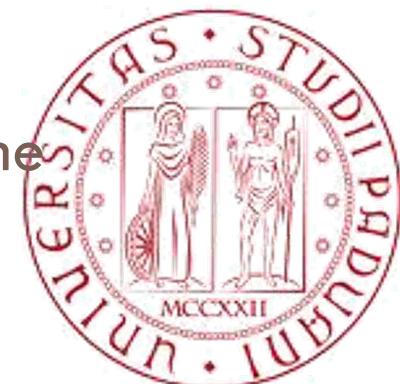


# Le origini dell'informatica

Prof.ssa Cinzia Pizzi

Dipartimento di Ingegneria dell'Informazione  
Università degli Studi di Padova





# Hilbert e il formalismo

- Alla fine dell'800 la matematica si interroga sui propri fondamenti
- David Hilbert propone 23 problemi matematici fondamentali, da risolvere nel corso del XX secolo





# Hilbert e il formalismo

- La matematica è un sistema formale completo?  
Ovvero posso dimostrare qualsiasi verità a partire dai suoi assiomi?
  
- Il problema della decisione: esiste un procedimento “meccanico” (passo-passo, in tempo finito) per decidere se una “proposizione” è o meno conseguenza logica di altre?



# Invece...

- Godel: Teorema di incompletezza (1931)
  - In un sistema formale, sufficientemente complesso da contenere l'aritmetica, **esistono delle “affermazioni” che non è possibile dimostrare ne’ vere ne’ false a partire dagli assiomi del sistema stesso**

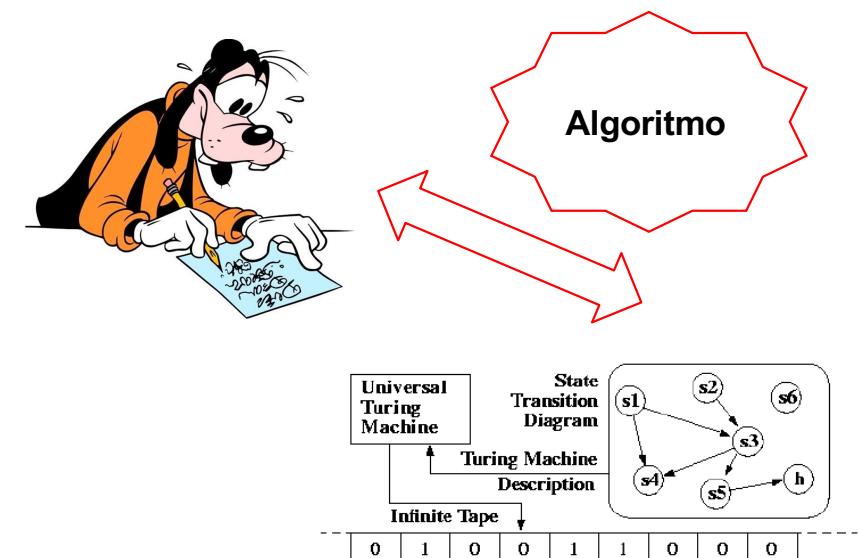


# E l'altra domanda?

- Church-Turing-Kleene: formalizzano la nozione di “metodo meccanico” con formalismi equivalenti
  - Il più famoso è la **macchina di Turing** (1936)
  - Tesi di Church-Turing: **tutto ciò** che è computabile è computabile da una macchina di Turing **universale**

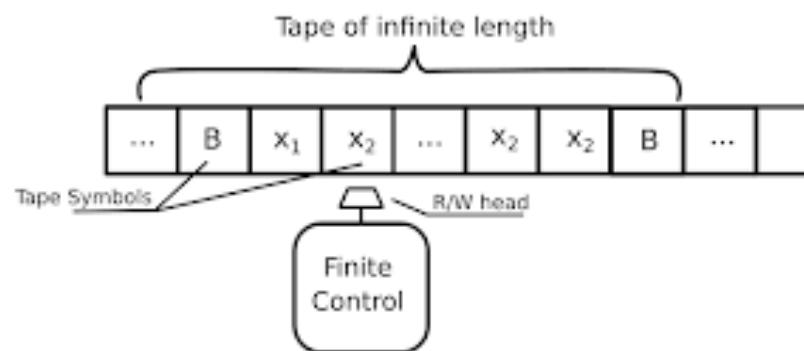


Fondamenti di Informatica



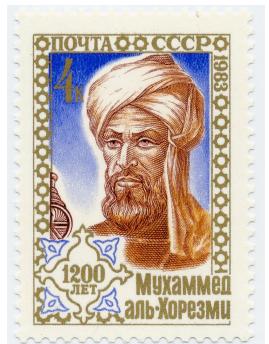
# Ma i computer moderni non sono “migliori”?

- I computer che conosciamo sono equivalenti in termini di **capacità computazionale** a una macchina di Turing universale
  - Possono però essere più o meno veloci, ma se un problema non e’ “risolvibile” con una TM, non lo e’ neanche con un supercomputer



# Algoritmo

- Si dice **algoritmo** la **descrizione** di un metodo di soluzione di un problema che
  - sia **eseguibile** (sequenza di passi eseguibili)
  - sia **priva di ambiguità**
  - arrivi a una conclusione con un **numero finito** di passi
- Un computer può risolvere soltanto i problemi per i quali sia noto un algoritmo



Muhammad ibn Musa **al-Khwarizmi** (c. 780 – c. 850)  
Matematico, astronomo, geografo persiano



# Dal problema all'algoritmo

- Esempio di problema numerico
  - Calcolo della media di n numeri
- Esempio di problema di ricerca di informazione
  - Contare il numero di occorrenze di una parola in un testo



# Esempio: media di n numeri

- Calcolare il punteggio medio dei frequentanti del corso di Fondamenti di Informatica al test di ammissione
- $\langle p_1, p_2, p_3, \dots, p_n \rangle$  sequenza di  $n$  numeri
- $(p_1 + p_2 + p_3 + \dots + p_n) / n$

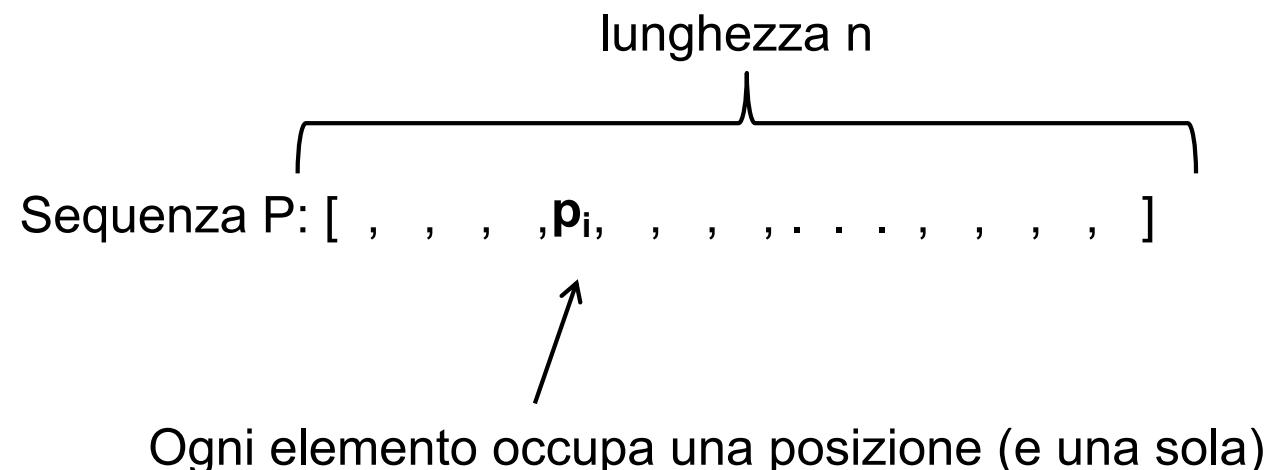
$$\frac{1}{n} \sum_{i=1}^n p_i$$

NB: definizione!

# Procedimento operativo

- Cosa significa e come si rappresenta una sequenza di numeri?

Le informazioni “atomiche” sono organizzate in un **Data container**





# Procedimento operativo

- Che tipo di numeri devo “manipolare”?

**NUMERI :**      **NATURALI**  
**INTERI**  
**RAZIONALI**  
**REALI**

# Operatori

- Quali operatori ho a disposizione? Ci sono criticità?
  - L'operatore “**Somma tutti**” non esiste in forma nativa!
  - L'operatore di divisione / può essere critico in funzione del valore del divisore

$$\frac{1}{n} \sum_{i=1}^n p_i$$

divisione      sommatoria



# Operatori

 $\sum$ 

Si può ridurre nella somma di più operatori a due operandi

=> **Scomposizione del problema**

$p_1 + p_2$



# Operatori

$\sum$

Si può ridurre nella somma di più operatori a due operandi

=> **Scomposizione del problema**

$$(p_1 + p_2) + p_3$$



# Operatori

$\sum$

Si può ridurre nella somma di più operatori a due operandi

=> **Scomposizione del problema**

$$((p_1+p_2) + p_3) + p_4$$



# Operatori

$\sum$

Si può ridurre nella somma di più operatori a due operandi

=> **Scomposizione del problema**

$$((....((p_1+p_2) + p_3) + \dots )+ p_{n-1}) + p_n$$

Ripeto => **ITERO**

**Utilizzo l'operatore di somma tra due operandi  
consumando i dati uno alla volta**



# Algoritmo



Se  $n == 0$  allora termina

Se  $n == 1$  restituisci  $p_1$

Somma i primi due numeri e mantieni il valore in un contenitore **Sum**

Usa un contatore **cont** per tenere traccia della posizione

**cont**  $\leftarrow 3$

Finche' **cont**  $\leq n$

1. Aggiungi a **Sum** il numero presente nella posizione **cont** della sequenza:  $\text{Sum} \leftarrow \text{Sum} + p_{\text{cont}}$

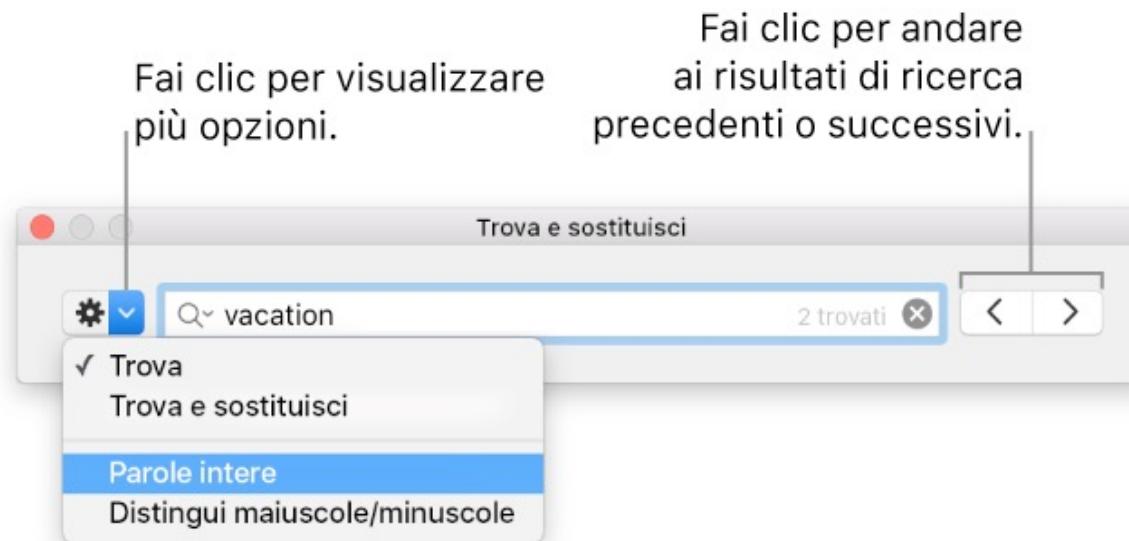
2. Incrementa **cont** di una unità:  $\text{cont} \leftarrow \text{cont} + 1$

Restituisci **Sum/n**



# Esempio Pattern matching

- Dato un *testo* e un *pattern*, trovare il numero di occorrenze di quel pattern nel testo

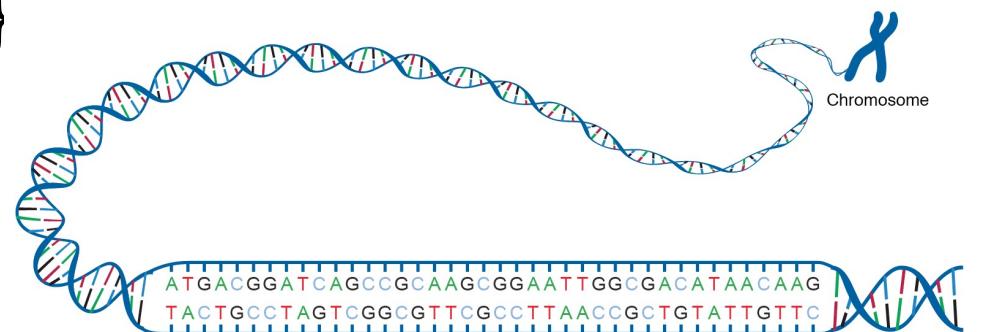


**Cos'è un testo?**

**Cos'è un pattern?**

# Formalizzazione

- Dato un alfabeto  $\Sigma$  sia il testo che il pattern sono definiti come sequenza finita di simboli dell'alfabeto, quindi il pattern non è necessariamente una “parola intera”.
  - Ad esempio: DNA= $\{A,C,G,T\}$
  - T = ACACCGTACC
  - P = ACC
  - Match: AC**ACC**GT**ACC**





# Formalizzazione

- Dato un alfabeto  $\Sigma$ ; m, n interi,  $m << n$
- Text:  $[t_1, t_2, t_3, \dots, t_n]$  ,  $t_i$  in  $\Sigma$
- Pattern:  $[p_1, p_2, \dots, p_m]$  ,  $p_i$  in  $\Sigma$
  
- Definiamo operatore per il confronto di caratteri:  
“=?” (vero/falso)



# Idea risolutiva

|    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|-----|
| A  | C  | A  | C  | C  | G  | T  | A  | C  | C   |
| t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 |
| =? | =? | =? |    |    |    |    |    |    |     |

|    |    |    |
|----|----|----|
| p1 | p2 | p3 |
| A  | C  | C  |

|    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|-----|
| A  | C  | A  | C  | C  | G  | T  | A  | C  | C   |
| t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 |
| =? | =? | =? |    |    |    |    |    |    |     |

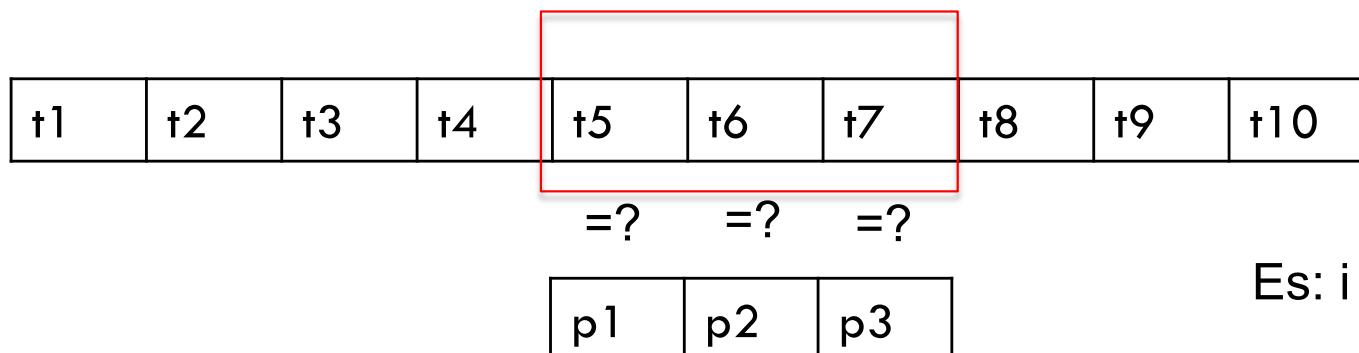
|    |    |    |
|----|----|----|
| p1 | p2 | p3 |
| A  | C  | C  |

|    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|-----|
| A  | C  | A  | C  | C  | G  | T  | A  | C  | C   |
| t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 |
| =? | =? | =? |    |    |    |    |    |    |     |

|    |    |    |
|----|----|----|
| p1 | p2 | p3 |
| A  | C  | C  |

# Ma in pratica?

- “Sposto il pattern” non ha senso in termini pratici
- Utilizzo due indici
  - Indice  $i$  per la posizione nel testo
  - Indice  $j$  per la posizione nel pattern
- Confronto per tutta la lunghezza del pattern partendo dalla posizione  $i$  del testo





# Algoritmo



Definisco un contatore ***cont*** per il numero di occorrenze: ***cont* <- 0**

Se  $n < m$  allora restituisci ***cont***

Per ogni posizione *i* del testo che mi permette di avere un match

*Definisco una variabile ***match*** (vero/falso) <- vero*

*Per ogni posizione *j* del pattern*

*se i caratteri corrispondenti  $t_{i-1+j}$  e  $p_j$  sono diversi  
***match*** <- falso*

*Se ***match*** == vero*

*incrementa ***cont****

**Restituisci *cont***



# A cosa servono gli algoritmi

- L'identificazione di un algoritmo è un **requisito indispensabile** per risolvere un problema (con il computer o senza)
  - La scrittura di un programma per risolvere un problema consiste, in genere, nella traduzione di un algoritmo in un qualche linguaggio di programmazione
- Prima di scrivere un programma, è **necessario individuare e descrivere un algoritmo**
- La definizione di algoritmi e la misura della loro efficienza è una parte importante dell'informatica (e anche di questo corso)



# Computational Thinking

## Cos'è il pensiero computazionale?

- ▣ Conoscenza di tecniche di astrazione e risoluzione di problemi algoritmici, proprie dell'informatica, e di come possano essere applicati in altre discipline
- ▣ Concettualizzare (= pensare a diversi livelli di astrazione)
- ▣ Complementa e combina il pensiero matematico e ingegneristico
  - Fondamenti matematici
  - Problem solving
  - Costruisce sistemi che interagiscono con il mondo reale



# Computational thinkers

## Concetti

- Logica
  - ▣ Predizione e analisi
- Valutazione
  - ▣ Esprimere giudizi
- Algoritmi
  - ▣ Esprimere la soluzione in un numero finito di passi
- Decomposizione
  - ▣ Ridurre in parti
- Astrazione
  - ▣ Rimuovere dettagli non indispensabili

## Approcci

- Tinkering
  - ▣ Cambiare le cose per vedere che succede
- Creatività
  - ▣ Progettazione e realizzazione
- Debugging
  - ▣ Trovare e correggere errori
- Perseveranza
  - ▣ non arrendersi
- Collaborazione
  - ▣ Lavorare assieme



## □ Risoluzione di problemi

- Quanto difficile è risolvere un problema (può dipendere dalla piattaforma e dai suoi limiti)
- Qual è il miglior modo per risolverlo (soluzione esatta/approssimata, accettiamo falsi positivi/negativi), etc

## □ Progettazione di sistemi

- Astrazione/decomposizione per progettare sistemi e gestire processi complessi
- Modellare aspetti rilevanti per rendere il problema trattabile / scegliere rappresentazione appropriata
- Progettare con modularità



# Take home message

- Gli algoritmi descrivono in modo non ambiguo la soluzione a un problema attraverso una sequenza finita di passi che porta al risultato in tempo finito
- Prima di scrivere un programma bisogna avere bene in testa l'algoritmo

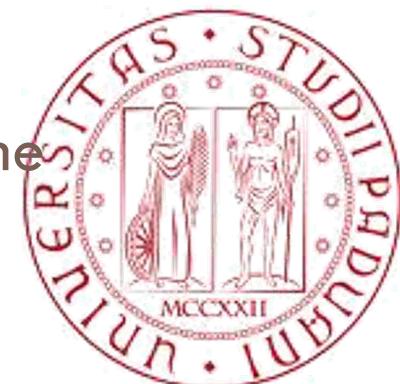




# I computer

Prof.ssa Cinzia Pizzi

Dipartimento di Ingegneria dell'Informazione  
Università degli Studi di Padova





# Obiettivi

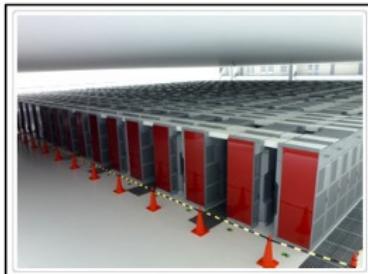
- Individuare le principali componenti di un computer
- Capire quali sono i rispettivi ruoli
- Studiare un modello di architettura di un computer

# Computers...

## Super Computers



Cray-1 (1979): 80 MHz, 8 MB RAM, 2.4 GB disks



K computer (2011)<sup>†</sup>: Fujitsu,  
705024 2GHz Sparc64  
processors

## Workstation/Desktop



Sun UltraSPARC Ili (2005):  
650 MHz, 2 GB RAM,  
80 GB HD



Intel Pentium 4 (2005):  
3.4 GHz, 4 GB RAM,  
300 GB HD

## Notebook



Intel Core i5 (2011):  
2.4 GHz, 8 GB RAM,  
500 GB HD

## Smart Phone/ Tablet



ARM-11 (2010):  
436 MHz, 128 MB RAM,  
16 GB Flash

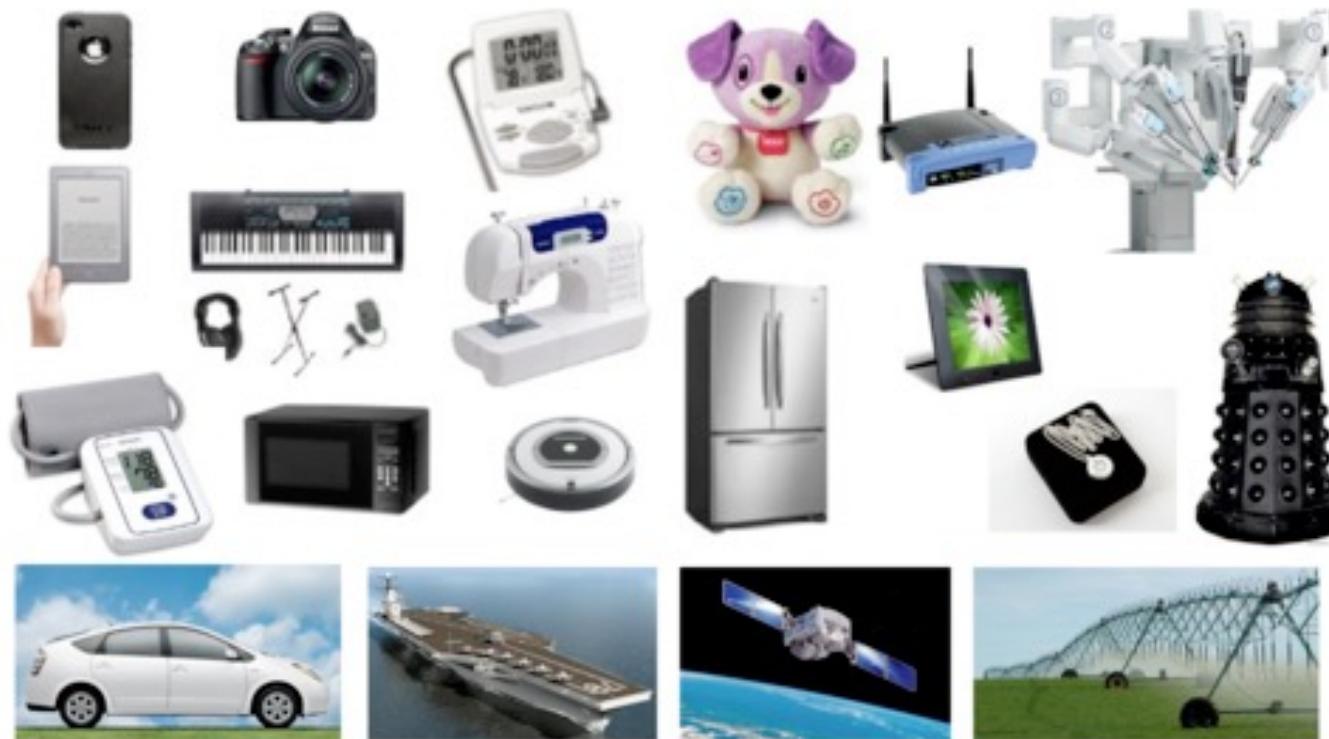


ARM Cortex A8 (2010):  
1 GHz, 256 MB RAM,  
64 GB Flash

# ...computers everywhere!

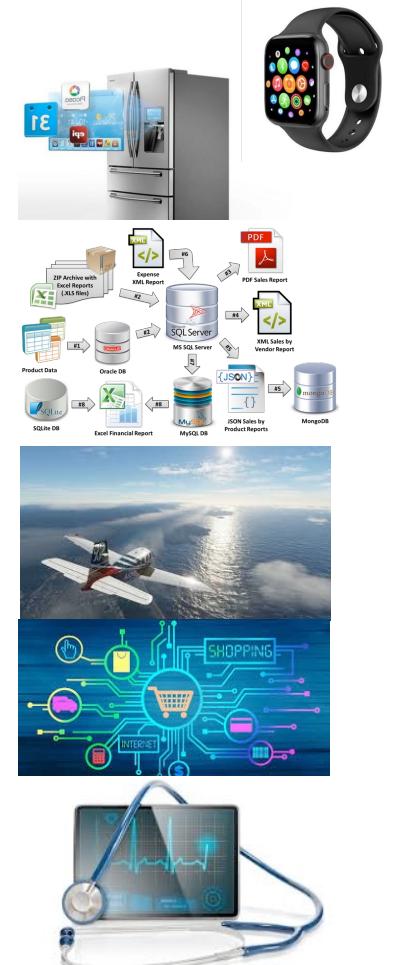
## SISTEMI EMBEDDED

Sistemi elettronici con “computer” integrato che svolge compiti pre-definiti non modificabili dall’utente



# Sull'uso dei calcolatori...

- A casa: elettrodomestici, immagini digitali nei televisori, dispositivi indossabili (smart watch)
- Sistemi informativi (basi di dati): supermercati, biblioteche, personale di un'azienda...
- Nel gioco: possono riprodurre con estremo realismo suoni e sequenze di immagini
- Nel commercio: pagamento elettronico
- Nelle applicazioni medicali
- ...



In realtà, tutto questo non è merito propriamente del computer, ma dei **programmi** che su di esso vengono eseguiti



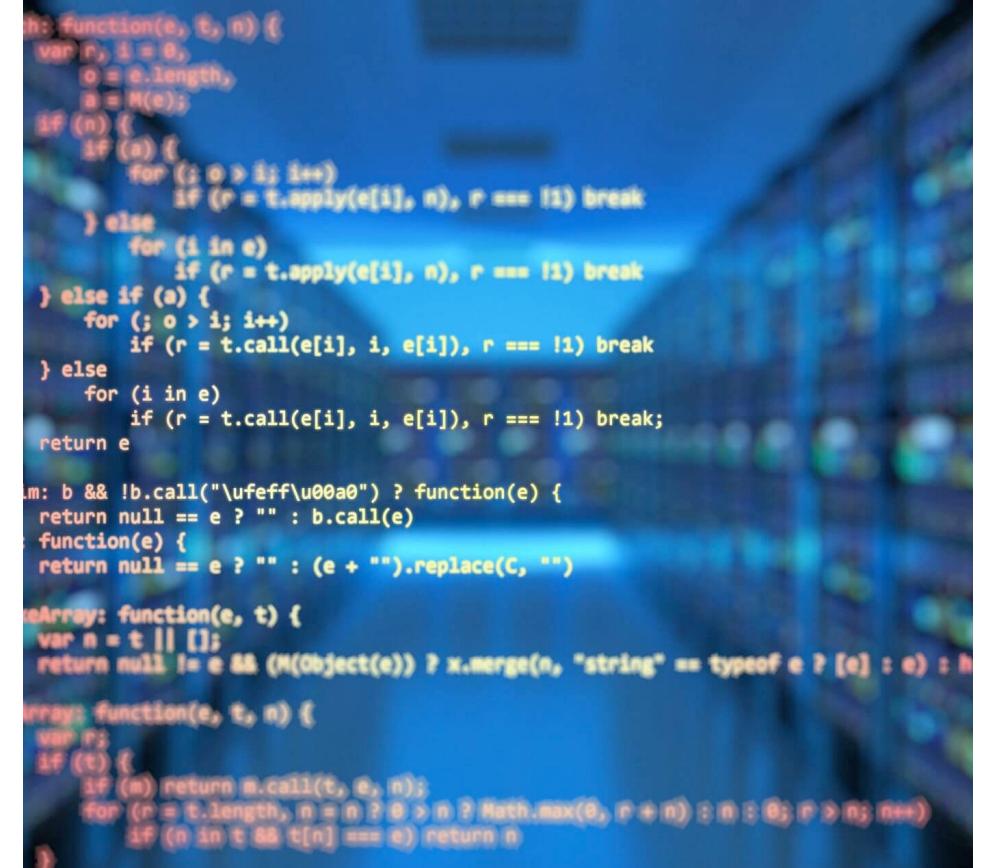
# Hardware e software

Un computer è un sistema per la memorizzazione ed elaborazione delle informazioni che opera sotto il controllo di una lista di istruzioni detta programma

**Hardware - parte fisica: elettronica,  
magnetica, ottica...**

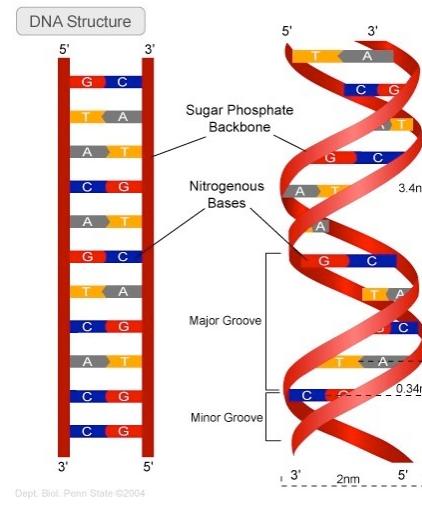
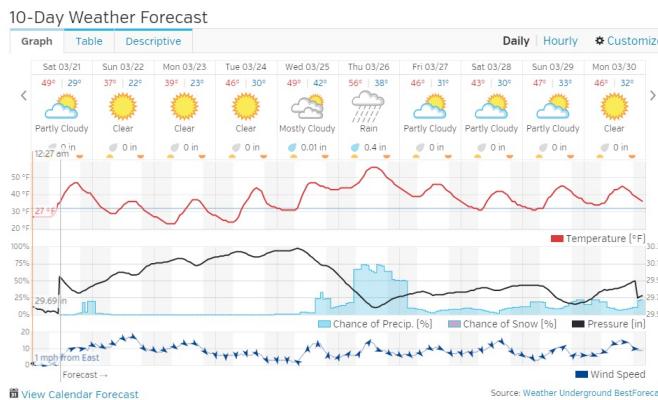


**Software – parte logica (informazioni):  
dati e programmi**



```
h: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        s = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r === !1) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], i, e[i]), r === !1) break;
    } else
        for (i in e)
            if (r = t.call(e[i], i, e[i]), r === !1) break;
    return e
}
m: b && !b.call("\ufe0f\ufe0f") ? function(e) {
    return null == e ? "" : b.call(e)
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "")
}
eArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string" == typeof e ? [e] : e) : h
)
}
array: function(e, t, n) {
    var p;
    if (t) {
        if (n) return m.call(t, e, n);
        for (r = t.length, i = n ? 0 > n ? Math.max(0, r + n) : n : 0; r > i; i++)
            if (n in t && t[n] === e) return n;
    }
}
```

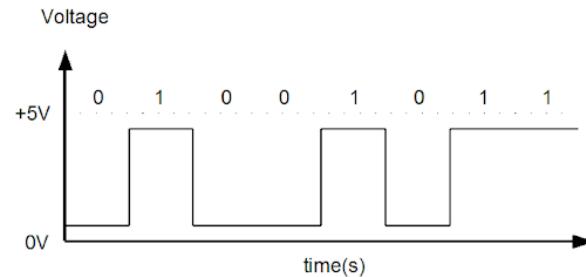
- I *dati* rappresentano qualunque tipo di informazione
  - Numeri, testi, suoni, immagini, filmati, ecc.



# Il mondo digitale



- Un **bit** è l'unità di misura dell'informazione
  - Due possibili valori: 0 o 1
  - Facile da rappresentare nel mondo fisico (on/off)



- Nei sistemi di computazione e comunicazione moderni **tutta l'informazione** viene rappresentata come sequenze di bit (stay tuned!)

- I **programmi** sono formati da insiemi di istruzioni (elementari) che vengono eseguite in un ordine stabilito

The image shows a code editor with two files side-by-side.

**PilaDiStringheTester.java (~/javaExamples/compito4-2019/)**

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5     int number, reverse = 0;
6     cout<<"Input a Number to Reverse: ";
7     cin>> number;
8
9     for( ; number!= 0 ; )
10    {
11        reverse = reverse *10 + number%10;
12        number = number/10;
13    }
14    cout<<"New Reversed Number: " << reverse;
15
16    return 0;
17 }
18 }
```

**def add5(x):**  
    **return** x+5

**def dotwrite(ast):**  
    nodename = getNodeName()  
    label=symbol.sym\_name.get(int(ast[0]),ast[0])  
    print ' %s [%s]' % (nodename, label),  
    **if** isinstance(ast[1], str):

```
61 // COMPLETARE LA CLASSE
62 class MiaPilaDiStringhe implements PilaDiStringhe {
63
64     int aSize;
65     String[] a;
66
67     public MiaPilaDiStringhe(){
68
69         aSize = 0;
70         a = new String[1];
71     }
72
73     public int size(){ return aSize;}
74
75     public boolean isEmpty(){return (aSize==0); }
76
77     public void push(String s){
78
79         if(s == null)
80             throw new IllegalArgumentException();
81
82         if(aSize == a.length){
```



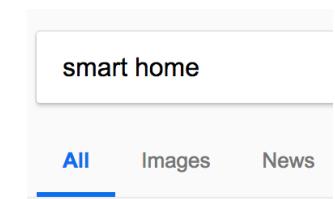
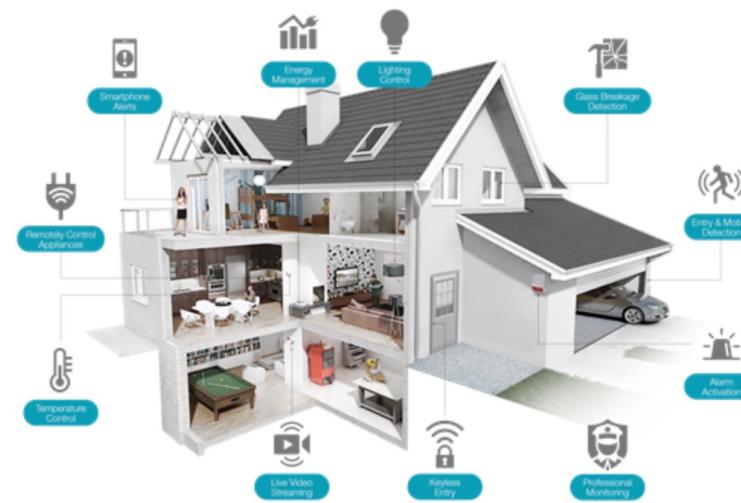
DIPARTIMENTO  
DI INGEGNERIA  
DELL'INFORMAZIONE

# Curiosita' sull'uso dei computer: fare previsioni e' "pericoloso"

**"Che bisogno ha una persona di tenersi un computer in casa?"**

**(Kenneth Olsen, fondatore della Digital Equipment Corporation (DEC), 1977)**

Successivamente Olsen ha ammesso di aver fatto questa affermazione, anche se specificando che le sue parole erano state prese fuori contesto: si stava infatti riferendo a computer costruiti per controllare le case, non ai PCs.



7,130,000,000 (2022)

# Curiosita' sull'uso dei computer: i primissimi "computer"





# Take home message

- Non esiste un solo tipo di computer
  - Notebook, desktop, super computer, smart phone, tablet, ...
- I computer sono tutti caratterizzati da
  - hardware (parte fisica)
    - Componenti elettronici, magnetici, ottici, ...
  - software (parte logica)
    - dati e programmi