

# INFORMATICA

Intelligenza Artificiale & Data Analytics

Precorsi a.a. 2023/'24

Docente: Gloria Pietropolli

# PRESENTAZIONI

**PIETROPOLLI GLORIA**

Dottoranda in Informatica - dipartimento di Matematica e Geoscienze

- ★ [gloria.pietropolli@phd.units.it](mailto:gloria.pietropolli@phd.units.it)
- ★ [github.com/gpietrop/PreCorso-INF-uniTS](https://github.com/gpietrop/PreCorso-INF-uniTS)
- ★ ufficio: C5 building

# OUTLINE DEL CORSO

## 20 ore di corso

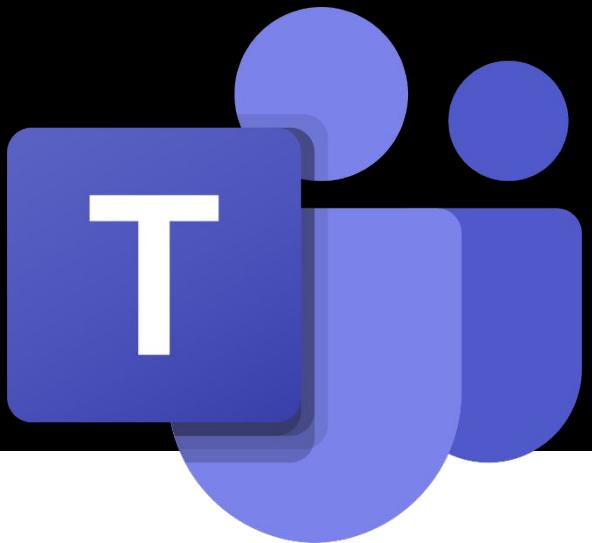
- inizio corso: 11 settembre 2023
- fine corso: 22 (forse 21?) settembre 2023
- lezione tutti i giorni dalle 14:00 alle 16:00
- inizio delle lezioni 14:15

*NON ESISTONO DOMANDE STUPIDE, L'UNICA COSA STUPIDA E' NON FARE DOMANDE!*

# PROGRAMMA DEL CORSO

- |  |    |
|--|----|
| 1. Computer: che cos'è e come parla        | 2h |
| 2. Il sistema operativo, file e filesystem | 3h |
| 3. Approfondimento sui file                | 3h |
| 4. Linux 101                               | 4h |
| 5. Version Control Systems e git           | 4h |
| 6. Basi di programmazione                  | 4h |

# TEAM DEL CORSO



I soli studenti già immatricolati possono essere aggiunti al Team del corso

Lì sarà possibile interagire con me e fare domande anche al di fuori delle lezioni

Link per richiesta di accesso al team

<https://cutt.ly/bW7nN64>



# 1. COMPUTER: CHE COS'È E COME PARLA

Definizione e nostro primo modello di  
computer

# CHE COS'È UN COMPUTER

«A computer is a machine that can be **programmed** to carry out sequences of **arithmetic** or **logical operations** **automatically**»



# CHE COS'È UN COMPUTER?

## Hardware

Parte materiale del sistema computer



## Software

Parte immateriale/logica del sistema computer

- Dati
- Programmi (istruzioni che il computer deve svolgere)

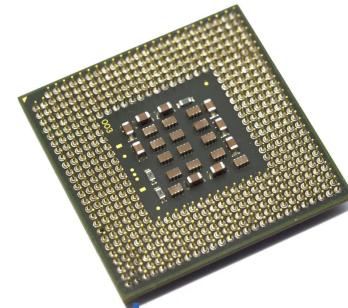
# HARDWARE - IL NOSTRO SCHEMA DI COMPUTER

**Unità di elaborazione (CPU):** esegue le istruzioni

**Memoria (di lavoro):** contiene

- Le istruzioni che il computer deve eseguire
- I risultati intermedi delle sue elaborazioni

Si svuota ad ogni riavvio del computer



**Bus:** connette tutte le diverse componenti

**Disco (fisso):** consente il salvataggio permanente delle informazioni



**Periferiche di input/output (I/O):** permettono di comunicare col computer

# 1. COMPUTER: CHE COS'È E COME PARLA

Come parlare ad un computer?

# LINGUAGGI NATURALI

Ciao,  
come va?



$$y = 5x^2 + \sin(x) + 10$$



# IL LINGUAGGIO MACCHINA

Le informazioni sono codificate in **linguaggio binario**.

Qualsiasi dato (un numero, una lettera, un'immagine) viene codificato tramite **sequenze arbitrariamente lunghe di 0 e 1**.

01001001 00100000 01101100 01101111 01110110 01100101 00100000 01000001  
01001001



I love AI

# CODIFICA BINARIA (I)

Nella **codifica binaria**, l'alfabeto è composto dai simboli **0** e **1**.

- Bisogna quindi trovare un modo per **convertire** ogni tipo di dato in sequenze di 0 e 1.

L'esempio più intuitivo è quello della **rappresentazione dei numeri naturali**, quindi partiamo da qua!

Il nostro sistema prevede di **rappresentare i numeri naturali in base 10**.

- Qual è **l'alfabeto** del sistema decimale? (**ma poi, cosa si intende per alfabeto?**)

# GLI ALFABETI

*“La serie ordinata di tutti i segni (o lettere) di cui una determinata lingua dispone per indicare il sistema di scrittura relativo ai suoni vocalici o consonantici (scrittura alfabetica).”*

- ❖ Alfabeto della lingua Italiana: **A, B, C, D...X, Y, Z**
- ❖ Alfabeto dei numeri in base 10: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- ❖ Alfabeto Musicale: **La-Si-Do-Re-Mi-Fa-Sol**
- ❖ **(e finalmente, quello che useremo noi)** Alfabeto dei numeri in base 2:
  - Abbiamo solo due simboli: **0 & 1**

# CODIFICA BINARIA (II)

Possiamo rappresentare un **numero naturale** nella **codifica binaria**

- ★  $0 \rightarrow 0$
- ★  $1 \rightarrow 1$  **HO FINITO I SIMBOLI DEL MIO ALFABETO BINARIO!**

**Ma posso usare più simboli dell'alfabeto insieme, no?**

- ★  $2 \rightarrow 10$
- ★  $3 \rightarrow 11$
- ★  $4 \rightarrow 100$

**Ma se ho un numero alto, devo elencare tutti i numeri precedenti per trovare il modo di scriverlo in alfabeto binario?**

**NO**

# REGOLA GENERALE PER LA CONVERSIONE

## METODO DELLE DIVISIONI SUCCESSIVE

- Divido il numero per 2
- Tengo traccia del **quoziente** e del **resto**.

### IL NUMERO 17

Riporto il  
quoziente

$$17/2 = \boxed{8}, \text{ resto } \boxed{1}$$

Tengo nota del  
resto

$$8/2 = 4, \text{ resto } 0$$

$$4/2 = 2, \text{ resto } 0$$

$$2/2 = 1, \text{ resto } 0$$

$$1/2 = \boxed{0}, \text{ resto } 1$$

1 0 0 0 1

Interrompo la catena quando il quoziente è 0

# ESERCIZIO

Rappresentare il numero 53 in codifica binaria.

# CONVERSIONE BINARIO → DECIMALE

**Ma ora devo anche capire come tornare indietro: se ho un numero scritto in base binaria come lo traduco in base decimale?**

Bisogna moltiplicare per due elevato alla posizione ogni cifra del numero (partendo da destra) ed infine sommarle

**Per fortuna è più facile a farsi che a dirsi!**

# CONVERSIONE BINARIO → DECIMALE

IL NUMERO 11001

$$\begin{aligned} 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 &= \\ 16 &+ 8 &+ 0 &+ 0 &+ 1 &= \\ 25 \end{aligned}$$

# ESERCIZIO

Rappresentare il numero **110101** in codifica decimale

# CODIFICA ESADECIMALE

**In informatica viene spesso utilizzata la base esadecimale** (ad esempio per l'indicizzazione delle «posizioni» della memoria)

**Vogliamo un alfabeto di 16 simboli, siccome noi abbiamo 10 cifre a disposizione, aggiungiamo 6 simboli rubandoli all'alfabeto**  
aggiungiamo le prime 6 lettere dell'alfabeto: **A B C D E F**

# CODIFICA ESADECIMALE

## (II)

Rappresentare un **numero naturale** nella **codifica esadecimale**

- ★  $0 \rightarrow 0$
- ★  $1 \rightarrow 1$
- ★  $10 \rightarrow 10$  **HO FINITO I SIMBOLI DEL MIO ALFABETO DECIMALE!**
- ★  $11 \rightarrow A$
- ★  $12 \rightarrow B$
- ★  $16 \rightarrow F$  **HO FINITO I SIMBOLI DEL MIO ALFABETO ESADECIMALE!**
- ★  $17 \rightarrow 10$
- ★  $18 \rightarrow 11$

# CODIFICA ESADECIMALE

## (II)

**Take Home Message:**  
posso creare un alfabeto di  
lunghezza arbitraria e con i  
simboli che voglio!

# PERCHÉ LA CODIFICA BINARIA?

I circuiti elettrici con cui è composto un computer hanno solo due stati di **tensione**:

- 0 = assenza di tensione
- 1 = presenza di tensione

**In che modo un computer è capace di memorizzare o di visualizzare delle informazioni con l'aiuto dell'elettricità?**

Lo stato di acceso o spento in cui può trovarsi un filo di corrente prende il nome di **bit**

→ rappresenta la più piccola quantità di informazione che può essere memorizzata su un qualsiasi computer

# IL BIT

*“Il bit (binary digit) è l’unità di misura standard dell’entropia, meglio nota come quantità di informazione”*



# IL BIT

Se si usano più fili di corrente si avranno a disposizione più bit.

→ Con più combinazioni di 0 e 1 sarà possibile rappresentare delle informazioni ancora più complesse:

- **1 filo di corrente = 1 bit = 2 stati** = spento oppure acceso = 0, 1;
- **2 fili di corrente = 2 bit = 4 stati** = spento oppure spento, spento oppure acceso, acceso oppure spento, acceso oppure acceso = 00, 01, 10, 11
- **3 fili di corrente = 3 bit = 8 stati** = 000, 001, 010, 011, 100, 101, 110, 111
- **4 fili di corrente = 4 bit = 16 stati** = 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

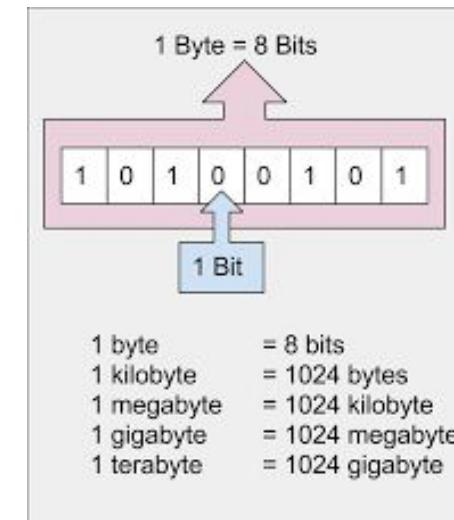
# IL BYTE

- In informatica, raggruppiamo i bit in **insiemi di 8**
- Un insieme di 8 bit è chiamato **byte**
  - a. La sigla del bit nel SI è "**b**"
  - b. La sigla del byte è "**B**".
- Il byte è l'unità di misura fondamentale per la memoria



Hobbit

Hobbyte



# RIASSUMENDO

- Il computer parla il linguaggio binario: 0 e 1
- Il **bit** è l'elemento base, può essere 0 o 1
- Un **byte** è un gruppo di 8 bit
- La memoria del computer utilizza elementi fisici (fori, carica elettrica) per rappresentare 0 e 1
- Memorie recenti usano semiconduttori, con la carica elettrica come indicatore del bit

# 1. COMPUTER: CHE COS'È E COME PARLA

La codifica dei numeri naturali e interi

# MA NON MANCA QUALCOSA?

- La codifica binaria dei numeri naturali ha un limite dovuto alla finitezza della memoria fisica.
- Non possiamo rappresentare infiniti numeri naturali, ma dobbiamo porre un limite massimo.

**Qual è il più grande numero che possiamo esprimere con 1 byte?**

- 1 byte = 8 bit
- ogni bit può contenere 0 o 1
- le possibili combinazioni sono  $2^8 = 256$
- possiamo rappresentare fino al numero **255**

**Ma soprattutto, non manca qualcosa? Cosa non stiamo rappresentando?**

I numeri Negativi (**Z**), i numeri Razionali (**Q**) e i numeri Irrazionali (**R**)

# CODIFICA PER $\mathbb{Z}$

## COME RAPPRESENTARE UN NUMERO NEGATIVO IN CODICE BINARIO

Un altro metodo per rappresentare numeri negativi è utilizzare il **bit del segno**

- **il bit più significativo (il primo a sinistra) rappresenta il segno del numero**
  - ◆ 0 per i numeri positivi.
  - ◆ 1 per i numeri negativi.
- **I restanti bit rappresentano il valore assoluto del numero**

a b b b b b b

Dedico il primo bit per determinare  
il segno del numero intero

I restanti 7 bit codificano i numeri  
da 0 a 127

# CODIFICA PER $\mathbb{Z}$

**Rappresentare il numero -34 in codifica binaria**

- Bit del segno (il primo bit a sinistra) è 1 perchè il numero è negativo
- Bit del valore assoluto rappresentano il valore assoluto del numero, cioè 34, ed è 100010
- -34 è rappresentato come 1100010 nel metodo a complemento di segno e magnitudine

# CODIFICA PER $\mathbb{Z}$

In realtà, questa non è la tecnica comunemente usata in informatica per rappresentare i numeri negativi, ma invece si usa un metodo chiamato: **complemento a due**

1. Rappresentazione binaria del valore assoluto del numero negativo
2. Inverti tutti i bit (0 diventa 1 e viceversa)
3. Aggiungi 1 al risultato ottenuto dalla fase 2

# CODIFICA PER $\mathbb{Z}$

**Rappresentare il numero -34 in codifica binaria**

- Rappresenta 34 in binario:
  - ◆ 34 in binario: 00100010
- Inverti tutti i bit (0 diventa 1 e viceversa):
  - ◆ Bit invertiti: 11011101
- Aggiungi 1 al risultato ottenuto dalla fase 3:
  - ◆  $11011101 + 1 = \boxed{11011110}$

# Q, R?

- ★ In informatica, dobbiamo rappresentare una vasta gamma di numeri reali, tra cui **numeri razionali** (fra le cui frazioni) e **numeri irrazionali** (come  $\sqrt{2}$ )
- ★ La rappresentazione di numeri reali può richiedere **approssimazioni** poiché alcuni numeri irrazionali hanno infinite cifre decimali
- ★ Durante i corsi di architetture degli elaboratori, vedrete **la rappresentazione in virgola mobile**

# CODIFICA DELLE STRINGHE [DI TESTO]

Oltre a rappresentare numeri, i computer devono gestire il **linguaggio naturale umano**

Il linguaggio naturale è rappresentato mediante la **mappatura di caratteri in numeri**

Il **codice ASCII** è uno standard comune che associa ogni carattere a un numero univoco

- Ad esempio, "HELLO" può essere rappresentato come [72, 69, 76, 76, 79]

**Quanti byte o bit servono per rappresentare la frase “Hello Joe”?**

- 9 bytes

# CODIFICA DELLE STRINGHE [DI TESTO]

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
00000000	0	Null	00100000	32	Spc	01000000	64	@	01100000	96	`
00000001	1	Start of heading	00100001	33	!	01000001	65	A	01100001	97	a
00000010	2	Start of text	00100010	34	"	01000010	66	B	01100010	98	b
00000011	3	End of text	00100011	35	#	01000011	67	C	01100011	99	c
00000100	4	End of transmit	00100100	36	\$	01000100	68	D	01100100	100	d
00000101	5	Enquiry	00100101	37	%	01000101	69	E	01100101	101	e
00000110	6	Acknowledge	00100110	38	&	01000110	70	F	01100110	102	f
00000111	7	Audible bell	00100111	39	,	01000111	71	G	01100111	103	g
00001000	8	Backspace	00101000	40	(	01001000	72	H	01101000	104	h
00001001	9	Horizontal tab	00101001	41	)	01001001	73	I	01101001	105	i
00001010	10	Line feed	00101010	42	*	01001010	74	J	01101010	106	j
00001011	11	Vertical tab	00101011	43	+	01001011	75	K	01101011	107	k
00001100	12	Form Feed	00101100	44	,	01001100	76	L	01101100	108	l
00001101	13	Carriage return	00101101	45	,	01001101	77	M	01101101	109	m
00001110	14	Shift out	00101110	46	.	01001110	78	N	01101110	110	n
00001111	15	Shift in	00101111	47	/	01001111	79	O	01101111	111	o
00010000	16	Data link escape	00110000	48	0	01010000	80	P	01110000	112	p
00010001	17	Device control 1	00110001	49	1	01010001	81	Q	01110001	113	q
00010010	18	Device control 2	00110010	50	2	01010010	82	R	01110010	114	r
00010011	19	Device control 3	00110011	51	3	01010011	83	S	01110011	115	s
00010100	20	Device control 4	00110100	52	4	01010100	84	T	01110100	116	t
00010101	21	Neg. acknowledge	00110101	53	5	01010101	85	U	01110101	117	u
00010110	22	Synchronous idle	00110110	54	6	01010110	86	V	01110110	118	v
00010111	23	End trans. block	00110111	55	7	01010111	87	W	01110111	119	w
00011000	24	Cancel	00111000	56	8	01011000	88	X	01111000	120	x
00011001	25	End of medium	00111001	57	9	01011001	89	Y	01111001	121	y
00011010	26	Substitution	00111010	58	:	01011010	90	Z	01111010	122	z
00011011	27	Escape	00111011	59	;	01011011	91	[	01111011	123	{
00011100	28	File separator	00111100	60	<	01011100	92	\	01111100	124	
00011101	29	Group separator	00111101	61	=	01011101	93	]	01111101	125	}
00011110	30	Record Separator	00111110	62	>	01011110	94	^	01111110	126	~
00011111	31	Unit separator	00111111	63	?	01011111	95	_	01111111	127	Del

# RIASSUMENDO

- I numeri naturali e interi possono essere convertiti in rappresentazioni binarie nei computer.
- Gli interi usano n bit (16 o 32), dove il primo bit indica il segno (+ o -) e i restanti (n-1) rappresentano il valore assoluto.
- I caratteri di testo sono associati a numeri tramite una mappa, come il codice ASCII.
- Questa mappatura permette di rappresentare simboli e caratteri utilizzando numeri naturali univoci.

# 1. COMPUTER: CHE COS'È E COME PARLA

Comunicare con un computer

# VERSO LA PARTE SOFTWARE...

## Hardware: La Parte Fisica

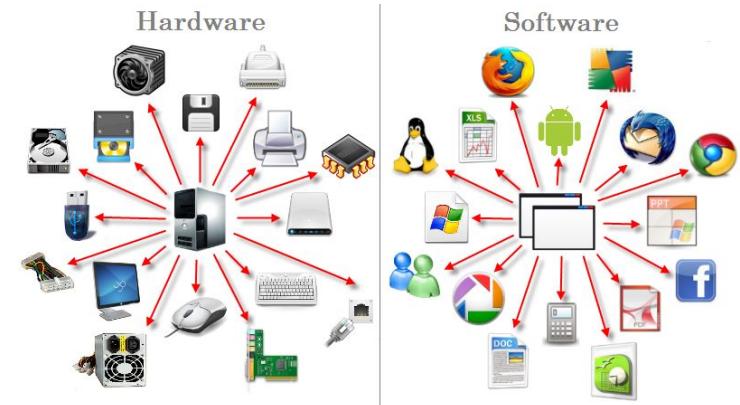
- L'hardware è la parte fisica di un computer, composta da componenti materiali
- Processori, schede grafiche, periferiche di input/output...

## Software: La Parte Immateriale

- Il software è la parte immateriale di un computer
- Comprende il sistema operativo, applicazioni, script, dati memorizzati...
- Guida il comportamento dell'hardware e consente al computer di eseguire varie attività.

## Interazione tra Hardware e Software

- L'hardware e il software lavorano in sinergia:
- L'hardware esegue le istruzioni del software
- Il software sfrutta l'hardware per svolgere compiti specifici



# PROGRAMMAZIONE

«A computer is a machine that can be **programmed** to carry out sequences of arithmetic or logical operations automatically»

La parte di **programmazione** si riferisce al fatto che noi possiamo **istruire il computer a svolgere determinate sequenze di operazioni (istruzioni, programma)**.

# COME COMUNICARE?

Abbiamo visto inoltre come un computer è in grado di rappresentare, utilizzando il linguaggio binario, dati come numeri o stringhe di testo.

Come possiamo comunicare al computer questi dati?

Come possiamo istruirlo a manipolare questi dati e a produrre una risposta?

Esempio: voglio istruire un computer a restituire il risultato della somma di due numeri  $a$ ,  $b$  e dividere questa somma per un terzo numero  $c$

Devo imparare il linguaggio binario?



Come comunico i valori di  $a$ ,  $b$ ,  $c$ ?

Come dico al computer di fare la somma di  $a$ ,  $b$  e di dividere per  $c$ ?

# FORTUNATAMENTE, NO

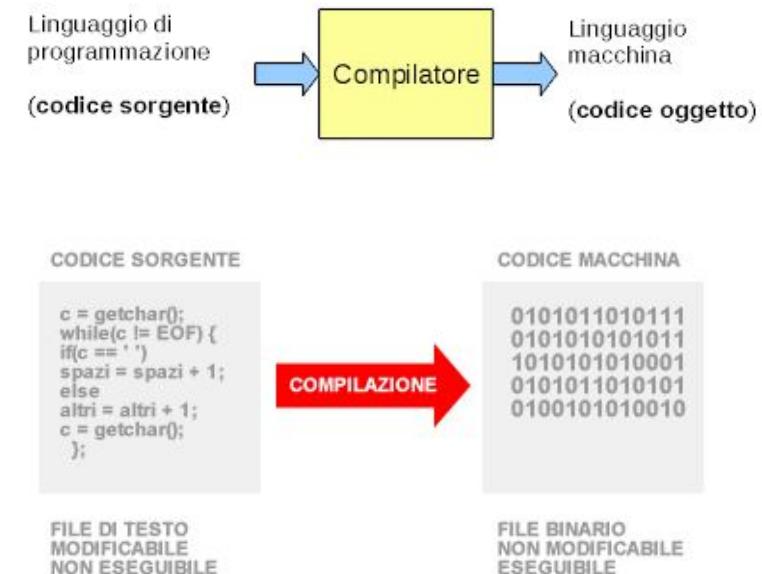
Sin dagli anni '50, l'obiettivo è stato trovare modi per consentire agli esseri umani di comunicare con i computer senza dover utilizzare rappresentazioni binarie complesse

Si scrive una sequenza di istruzioni - **codice** - in un formato che sia comprensibile agli esseri umani

- Questo codice è scritto in **linguaggi di programmazione** progettati per essere vicini alla comprensione umana

Successivamente, un programma speciale chiamato **compilatore** o **interprete** traduce il codice scritto dall'umano in linguaggio binario, che è comprensibile per il computer.

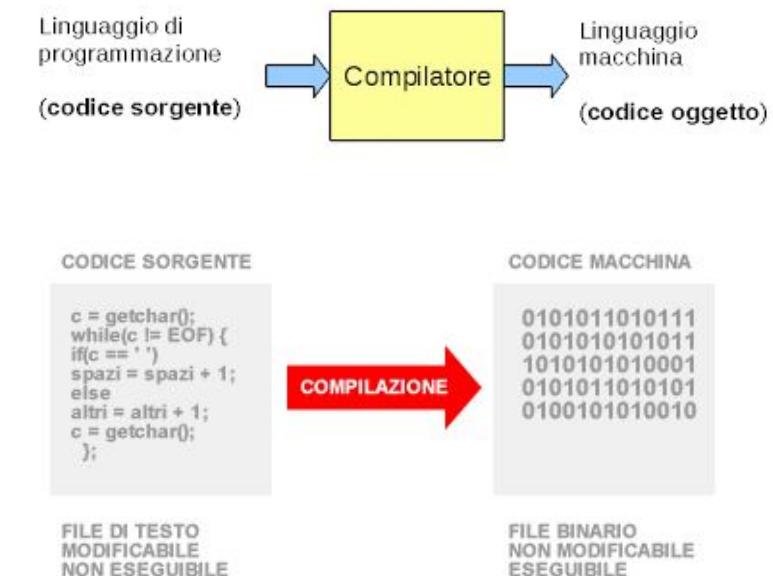
- Questo processo consente ai computer di eseguire le istruzioni scritte dagli esseri umani.



# FORTUNATAMENTE, NO

Ora non rimane che capire come scrivere questi programmi per poter comunicare con il compilatore

Ma, cosa è un programma? Cosa è un codice? Quanti modi esistono di comunicare con il Computer?????



# MA, CHE COS'È UN PROGRAMMA?

*“Un programma, in informatica, è un  
procedimento algoritmico applicato ad  
un problema dato da automatizzare”*

# ALGORITMO

*“Un algoritmo è una strategia che serve per risolvere un problema ed è costituito da una sequenza finita di operazioni (dette anche istruzioni)...”*

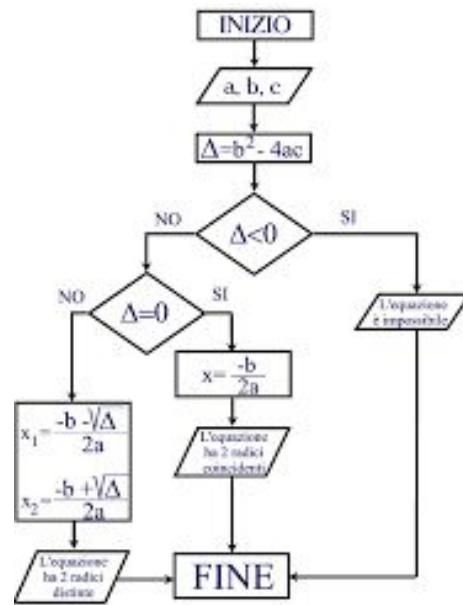
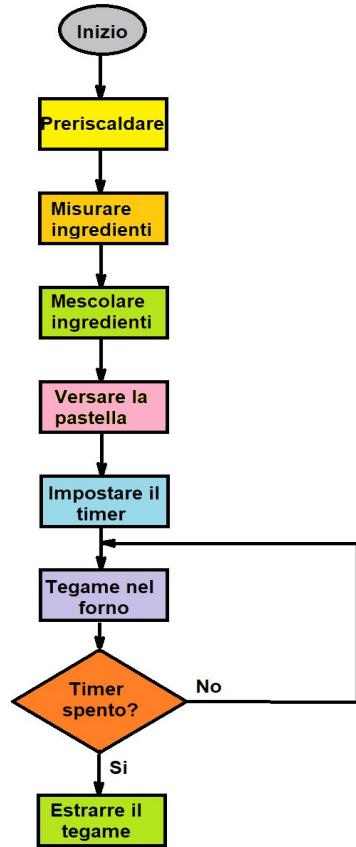
# ALGORITMO

Gli algoritmi sono:

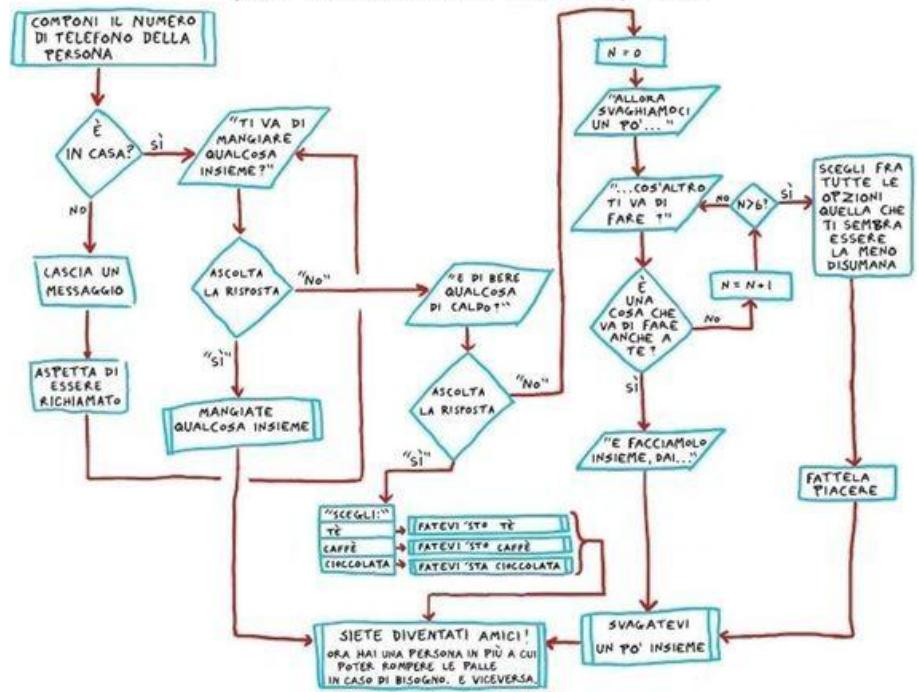
- **Ben definiti**: hanno passaggi chiari e precisi
- **Finiti**: si completano in un numero finito di passaggi
- **Efficaci**: risolvono il problema in modo efficiente

Gli algoritmi svolgono un ruolo fondamentale nella programmazione e nell'informatica, poiché guidano il comportamento del computer

# ALGORITMO



## L'ALGORITMO DELL'AMICIZIA DEL DR. SHELDON COOPER, Ph.D



# DI NUOVO AL PROGRAMMA

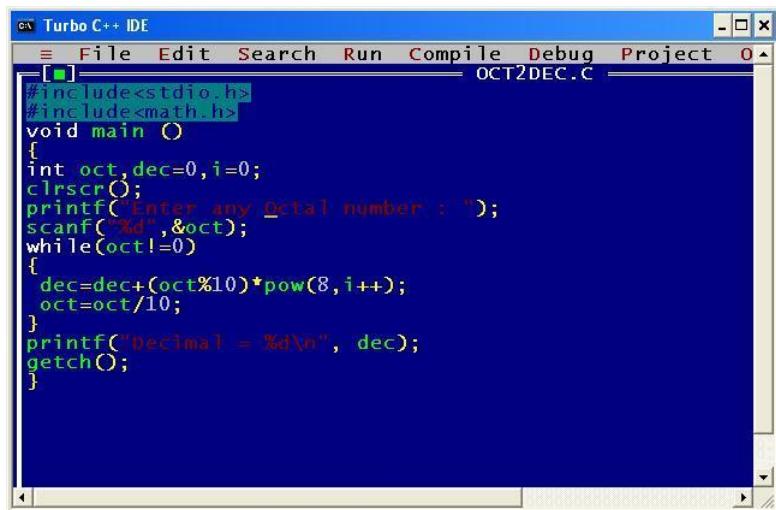
Possiamo quindi vedere il programma come un insieme di istruzioni esprimibili secondo un algoritmo

Il programma è un insieme di **linee di testo**, detto **codice**

*“Il codice indica una sequenza ordinata di comandi che possono essere convertite in linguaggio macchina e successivamente eseguite su un computer”*

# COMPILAZIONE (SEMPLIFICATA)

Programma scritto  
in linguaggio di  
programmazione



```
#include<stdio.h>
#include<math.h>
void main()
{
    int oct,dec=0,i=0;
    clrscr();
    printf("Enter any octal number : ");
    scanf("%d",&oct);
    while(oct!=0)
    {
        dec=dec+(oct%10)*pow(8,i++);
        oct=oct/10;
    }
    printf("Decimal = %d\n", dec);
    getch();
}
```

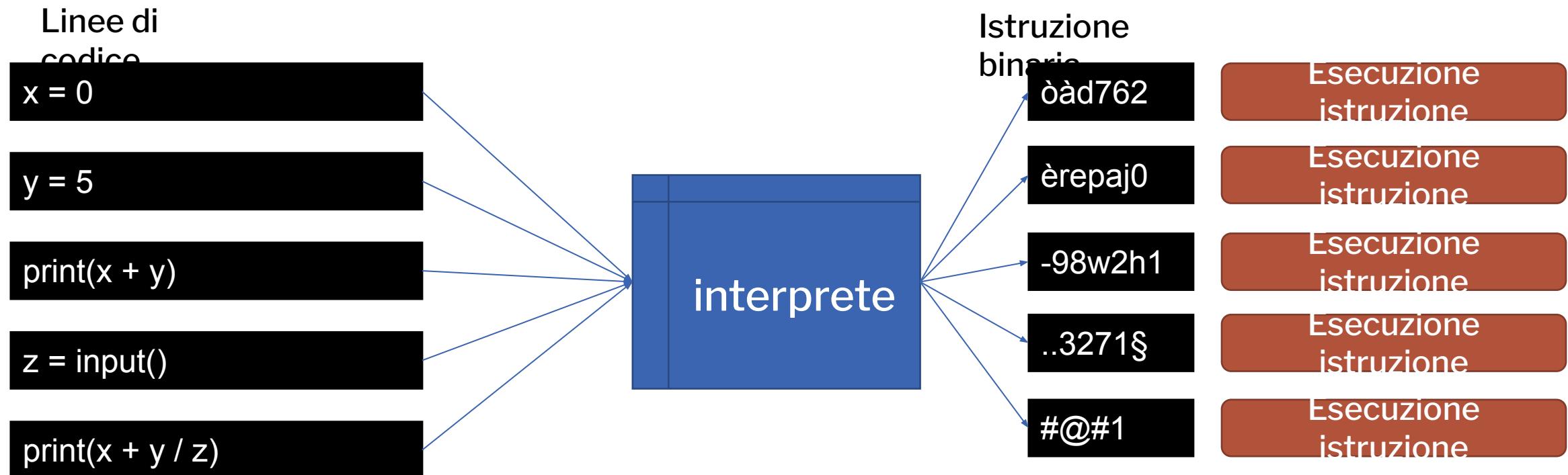
compilatore

Istruzioni in  
linguaggio  
macchina

```
%PNGCRLF
SUBLF
NULNULNULCR
IHDRNULNULSOHöNULNULSOHdBSSTXNULNULNULYö^zNULNULNULSOhsRGBNULöIFS
y—N,?yT[NULNULhöÖäë_yúÖöp,,#FSäBSNULOENOü/ö[çö[àö×4ñö.ż!CAN·xöä"
%y·ixäfälöfgzöçESCVFdZ;8Üçú^o_Ýicí...gDC2ACKLF
ç<ö[müÜöyöö÷³A$ÜöY7løyö*x<=DçÜñoÜöiØ/k>öU“Æf1~Ü<0üö”ýÜj—ü|“þöz
ðææùDC37%é=-jjiëtiÅ/€•äøVIIUSöSTXöö³czæöc ` US/FS{ùÿÅ# ð~9g>3ðåÓ!
!pWAEGOT|2ME^€;l.,í“Æ’-DLEøe"p^Di/%äöaETXACK\SI\DC1KV\ð—øACK,NU
þWxxi<_STX\÷ÜK_ýEOT&STXöI«DC3ACK†þÖBSNULCö ðUSZl>Üyö... yVcoÙACI
µW6VfÖö²ÜSK;:«é»µÍ•=ÄCR
íU-C’Zäöåäf®_cSUB...äyBSÜg6öö5GS&m»YxæÀ{=ö@ö9FSvÆ=³>zÄ°mp—
üP8Eäö07t‰$öliÄärBöðüäxÄ4}¹—ÄSUBüäž-qACKÖö%, ^~XÜ~xžáöcæ+, fc“EOT
ðäNvñIAÜtö-, .m=F1#ÖZBjäÄSYNIdSöga1dC1ÄVBSbçxRS-4iWiä-ööÉ-íöiÜJUSR
z'fÁ) gDLE&é:K-JuäžAuPjEMETBIfæætÜüæGG{@í}T+9ZNAK["ämGrÖCANJñ`N1,
.i|ç‘STX’ ²äöFSñ[Hgq«öüV’ ~tíDC4gá"SYNFsØ,A=í\ñCANRS(eéíéÉž-a±[YÉ
USØENOfmÖnëQ)aFFFEC1ë_w» mÉiäSMcZurÖSYüä, of/ FSµMm³ajB;, taDC4
ñZ%[ÄpÄöYStISOWRS18An~SUBEö8|Äp9ACKé1, ñmÖöQ!xg>DC4ESCACKéiüü±
ÉyäçÝSOH«ÍséÄ/EïÖB,Sý£ñi{éYäETX,¹NULxDö-jö-z4DLEöAö»äöi...\ÜSOH4è1H
z”>SéÑNSOHEFBSB, åDC26cz Jö-³%9o| aq1ga•1»Ü58^x5PöÖ”pD-ÜÜ”2<YNULf>J
f~A””ööé, è~şEopC-ÖÜç;Z»EMatzAWGåÈPÍLÖÄ/ËA|ÖöBöZ+XçEú1BEL¹DLEö2CzU
iiDC2AçÑåPSOHöBS,, lBSñACKQ#USFSACK=öZfEëSBf10_NULE”xÜGSBS()“DC3usf
...Y-7ë9LF
I`/ö-öö-USþpmöSöQsETB”Æ)@LIF
U-ÑØkNUL#¥jöëšq, €üuëlÿééëå, t¡ | RS§UofBSMDMDLEÜçöLíëSYN·#ó‡;+‘À$j¥
REOTxöh@!+, ,ÈÖ*%CR
```

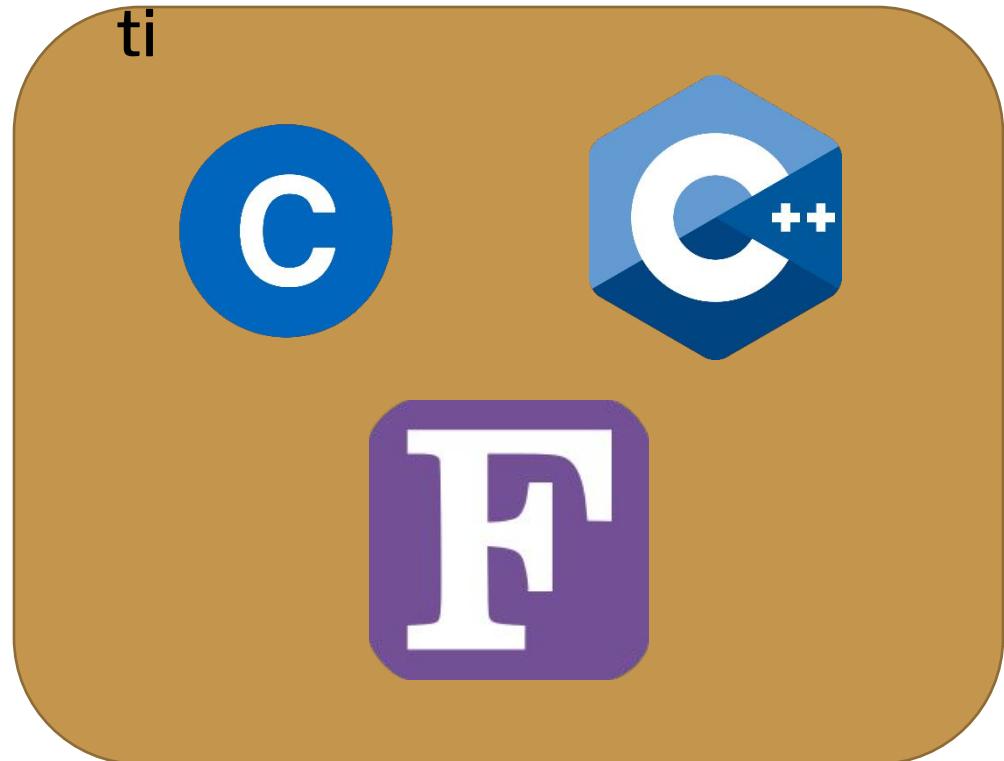
«FILE  
ESEGUITILE»

# INTERPRETAZIONE (SEMPLIFICATA)

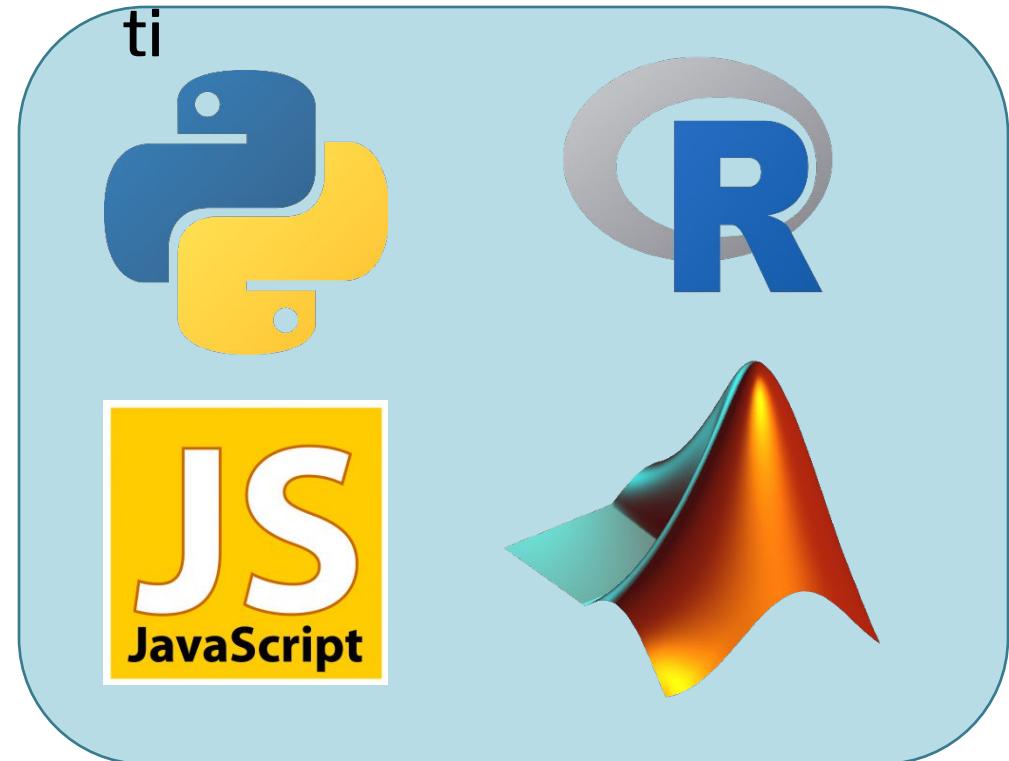


# LINGUAGGI COMPILATI E INTERPRETATI

Compila  
ti



Interpreta  
ti



# ISTRUZIONI E OPERATORI LOGICI

Nella sua accezione più di base, il programma è costituito da istruzioni che possono essere:

- Di tipo **matematico** (es. somma due numeri, fai la radice di questo numero...)
- Di tipo **logico** (es. si verificano questo E quest'altro, NON si verifica questo...)
- Di tipo **imperativo** (es. SE si verifica qualcosa ALLORA fai questo)

# ESECUZIONE DEL PROGRAMMA

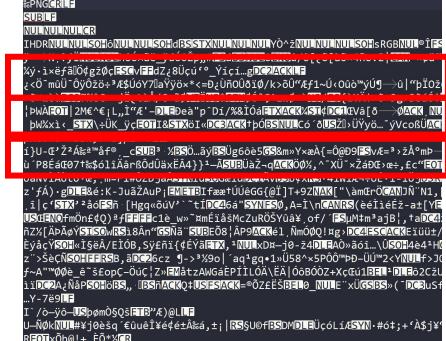
Si dice che il computer **esegue** un programma quando ne svolge le istruzioni contenute nel codice (compilato o interpretato).

La fase di esecuzione può essere crudamente così riassunta:

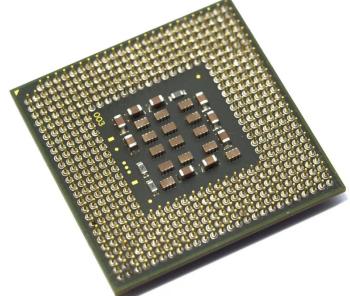
Istruzioni in linguaggio  
macchina contenute su disco



«Isolamento» di  
istruzioni «atomiche»



Passaggio  
dell'istruzione  
all'elaboratore che  
effettua  
l'esecuzione vera e  
propria  
dell'istruzione



# OPEN-SOURCE E CLOSED-SOURCE

## **Closed-source: il codice del programma è nascosto**

Viene rilasciato solo l'eseguibile

Non si sa effettivamente che cosa faccia il programma una volta in esecuzione

## **Open-source: il codice è accessibile**

Open source ≠ gratis

In alcuni casi è possibile apportare modifiche al programma in maniera collaborativa

# RIASSUMENDO

- ★ Un **algoritmo** è un metodo o un procedimento per risolvere un problema che coinvolge una serie finita di operazioni
- ★ Un **programma** è un insieme finito di istruzioni espresse in un linguaggio di programmazione, generalmente sotto forma di codice sorgente
- ★ Le **istruzioni in un programma** possono essere di varia natura, inclusa matematica, logica, gestione dei dati e altro ancora.
- ★ L'**esecuzione di un programma** coinvolge il caricamento del programma in memoria, l'isolamento delle operazioni in istruzioni atomiche e l'esecuzione di queste operazioni da parte della CPU.

# LA PROSSIMA VOLTA...

Inizieremo a introdurre:

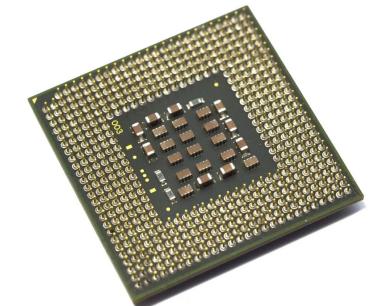
1. Il sistema operativo
2. La nozione di file
3. L'organizzazione dei file in cartelle gerarchiche e la nozione di *filesystem*

Portate un PC se ne avete uno a disposizione, ci saranno dei momenti interattivi.

# HARDWARE: UNITÀ CENTRALE

Il «computer» vero è proprio (la parte che «elabora i dati») è la cosiddetta **unità centrale**, che è composta da:

- Un processore o CPU (Central Processing Unit) che è la componente che fisicamente effettua i calcoli
- Una memoria volatile o RAM (Random Access Memory) che permette il salvataggio temporaneo dei dati elaborati dalla CPU

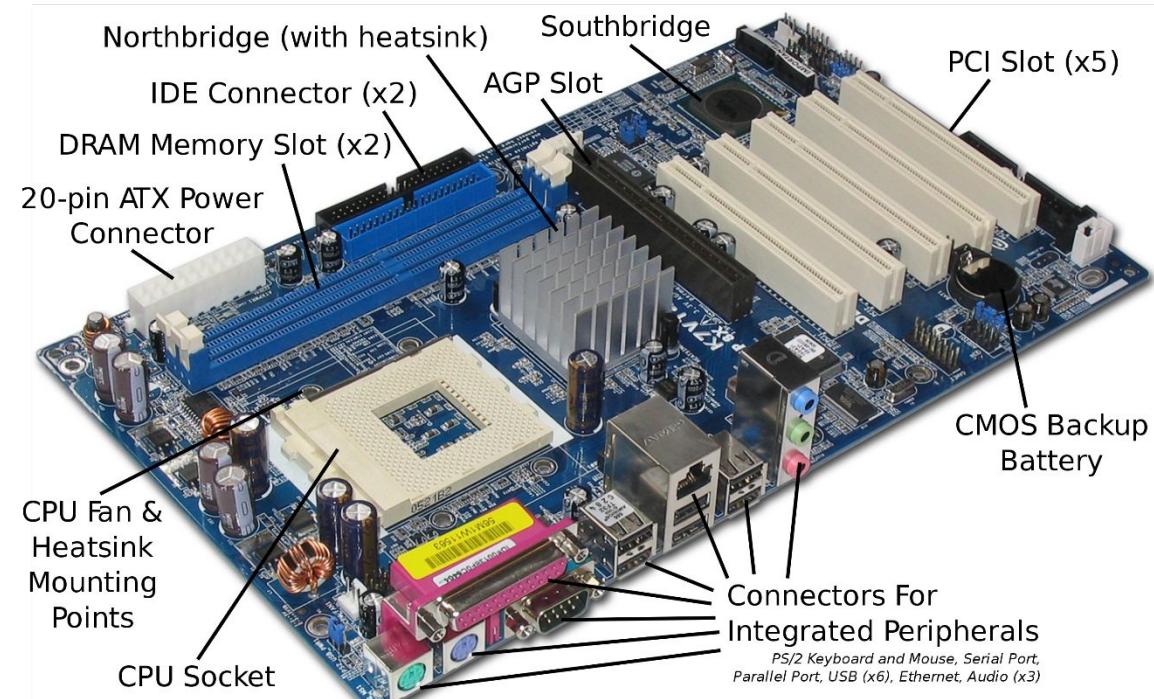


# LA SCHEDA MADRE

La scheda madre o MoBo (MotherBoard) fa anch'essa parte dell'unità centrale e permette il collegamento e l'interfaccia dei componenti finora descritti.

La MoBo permette inoltre il collegamento della scheda con l'alimentazione elettrica.

Manca ancora qualcosa?



# PERIFERICHE DI INPUT E OUTPUT (I)

Teoricamente, un computer con le componenti finora descritte potrebbe funzionare potenzialmente all'infinito.

Il problema è che:

- Non abbiamo modo di comunicare con il computer (**input**)
- Il computer non ha modo di comunicare i suoi risultati a noi (**output**)

La quasi totalità delle componenti finora non descritte ricade nella categoria delle **periferiche di input o output (I/O)**.

# PERIFERICHE DI INPUT E OUTPUT (II)

Sapete nominare 3 periferiche di input e 3 di output?

INPUT

OUTPUT

# PERIFERICHE DI INPUT E OUTPUT (III)

INPUT



OUTPUT



# EXCURSUS TASTIERE



Di solito possiamo classificare una tastiera in base ai primi 6 caratteri alfabetici della seconda riga

Le tastiere utilizzate in US e Italia sono di tipo QWERTY.

- Nonostante le lettere siano al medesimo posto, le tastiere IT e US differiscono per posizione e tipologia di simboli e altri caratteri

In Europa generalmente si utilizzano altri layout:

- DE (e gran parte dell'Europa continentale) → QWERTZ; FR → AZERTY...

Vi è un dibattito tuttora in corso sull'effettivo vantaggio di QWERTY & co.

- Tuttavia, sostituire il formato richiederebbe un cambio notevole nelle abitudini degli utenti
- Per questo motivo, un paradigma non efficiente (ma non terribile) ma enormemente radicato nelle abitudini di specifici utenti, si può anche denominare «il QWERTY di [qualsiasi cosa]»

# PERIFERICHE DI INPUT/OUTPUT MEMORIA DI MASSA

Vi sono ancora ulteriori periferiche che possono svolgere entrambe le funzioni (chiamiamole *ibride*)

La memoria di massa è una memoria persistente che permette

- Al computer di salvare i dati temporanei presenti in RAM
- All'utente di fornire un input al computer tramite le informazioni contenute nel disco



# PERIFERICHE DI INPUT/OUTPUT MODEM

Il modem è anch'esso una periferica ibrida in quanto permette un flusso, alternativo rispetto alle memorie di massa, sia in entrata che in uscita



# ALTRÉ PERIFERICHE

Vi sono altre periferiche che sono di supporto a quelle finora indicate, ma che non sono necessariamente da classificare come input o output.



La scheda di rete permette al computer di interfacciarsi con il modem affinché possa inviare a e ricevere dati da quest'ultimo

La scheda video (GPU) offre supporto alla CPU permettendo di svolgere in maniera più rapida i calcoli necessari alla riproduzione delle immagini sui monitor. Si può quindi pensare alla GPU come una CPU con capacità ridotte.



# PERIFERICHE: NOTE CONCLUSIVE

Ci sono ancora altre periferiche non coperte in queste slide (es. schede audio).

Molte periferiche «di supporto» nei computer recenti si trovano integrate all'interno della scheda madre (audio, rete...) o CPU (scheda grafica).

# RIASSUNTO

Possiamo visualizzare un computer come una composizione delle sue componenti materiali (*hardware*) e immateriali (*software*).

La *base* del computer è l'*unità centrale*, composta da CPU (calcolatore in senso stretto), memoria [RAM] e scheda madre.

Le altre componenti sono usualmente

- Input e/o output (I/O)
- Periferiche di supporto alle componenti I/O



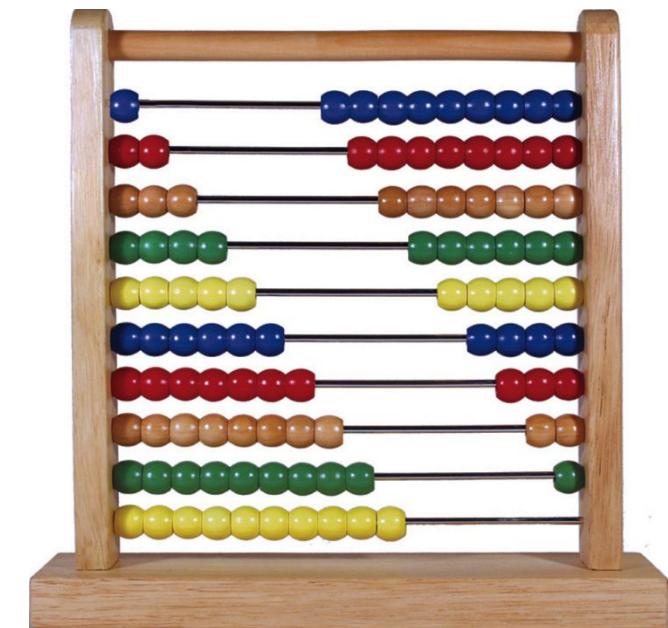
# STORIA DEL COMPUTER

Excursus storico

# CHE COSA PUÒ ESSERE EFFETTIVAMENTE DEFINITO «COMPUTER»?

Nel corso della storia, gli umani hanno spesso necessitato di macchinari per l'ausilio di calcoli basilari.

Se accettiamo qualsiasi supporto fisico per il calcolo come «computer», allora probabilmente il primo computer della storia è l'abaco, utilizzato da più di 4000 anni in Cina e Mesopotamia.



# IL PRIMO «VERO» COMPUTER

L'abaco però è solamente un *supporto*, che facilità l'umano nel compito di tenere i conti, ma non li effettua in autonomia.

Il meccanismo di Antikythera era un calcolatore meccanico per effettuare previsioni accurate sulla posizione di stelle, pianeti ed eclissi con anni di anticipo. Si stima sia stato costruito fra il II e il I secolo a.C.

I resti sono stati trovati nel 1902 all'interno del relitto di una nave romana scoperto anni prima a largo dell'isola ellenica di Antikythera.

Possiamo definirlo un computer vero e proprio perché, ad un input dell'utente, produceva un output (le posizioni astronomiche) conducendo autonomamente le computazioni necessarie.



# IL PRIMO COMPUTER MODERNO

Nel 1833, il matematico britannico Charles Babbage progettò (su carta) la macchina analitica, il primo progetto di **calcolatore programmabile** moderno.

Dati e istruzioni venivano passati sotto forma di schede forate. Anche l'output veniva prodotto in forma di schede forate, dacché la macchina era provvista di una «stampante» per la perforazione del cartone.

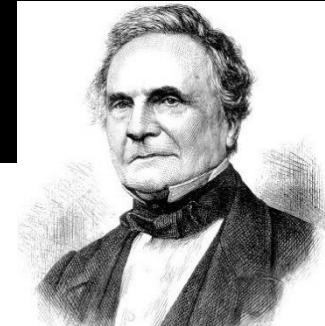
La RAM era composta da una serie di ingranaggi in grado di tenere 1000 numeri con 40 cifre decimali (ca. 17 kB di memoria).

La CPU era anch'essa composta da una serie di ingranaggi e le sue procedure erano *programmabili* tramite l'inserimento di pioli ad opera del programmatore. Poteva effettuare le operazioni  $+ - \times / \sqrt < > = \neq$ .

Nonostante gli impegni di Babbage, il computer non fu mai costruito in quanto il progetto era troppo avanzato rispetto alle tecnologie ingegneristiche dell'epoca.

Alcune varianti sono state costruite negli anni, l'ultima delle quali nel 1991.

La matematica britannica Ada Lovelace creò, nel 1843, dei programmi per la macchina analitica (utilizzando il linguaggio di programmazione a schede forate) ed è per questo definita il primo programmatore della storia.  
(NB: Ada Lovelace day → 12 ottobre '21)

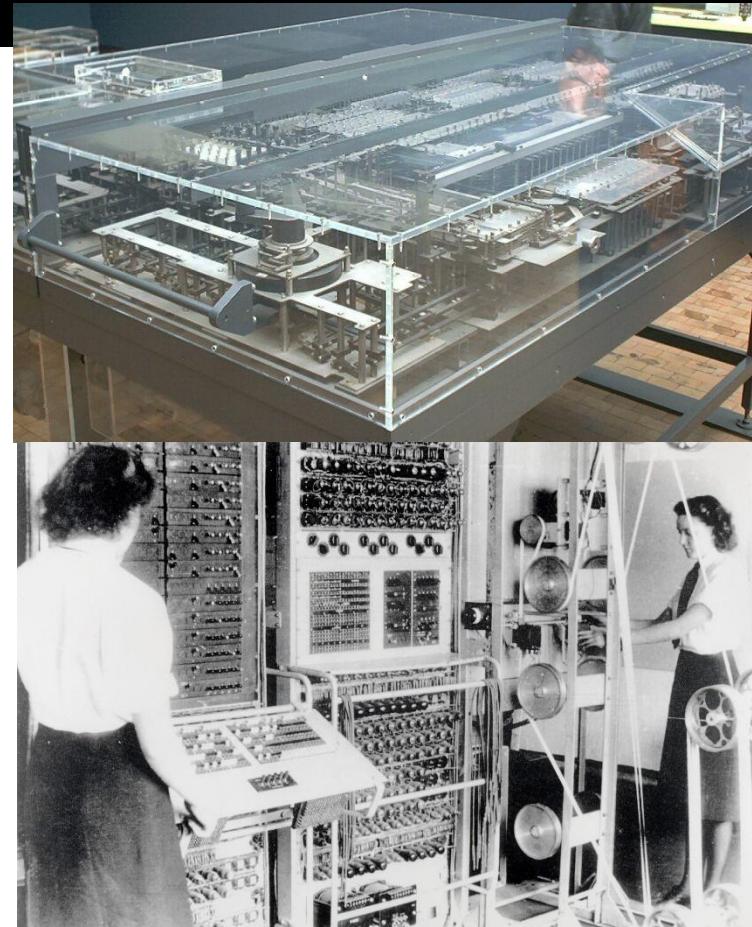


# I PRIMI COMPUTER FUNZIONANTI

Dagli anni '30 del '900, si iniziano a vedere investimenti notevoli per la costruzione del primo computer programmabile *funzionante*.

Il titolo va a Konrad Zuse, ingegnere tedesco, che nel '39 costruì lo Z1.

Nel '44, inoltre, il Colossus, costruito in Regno Unito da parte di un team guidato dal «padre dell'informatica» Alan Turing, portò alla decifrazione del codice Enigma, utilizzato dall'Asse per criptare i messaggi segreti. Questo evento contribuì notevolmente alla vittoria alleata nella II guerra.



# EVOLUZIONE DELL'I/O

Questi primi computer utilizzavano ancora l'idea «antica», già teorizzata da Babbage, di utilizzare schede forate per l'input.

Negli anni '50 si inizia ad utilizzare un monitor come periferica di output (anziché stampe o altre schede forate). La tastiera invece arrivò appena negli anni '70, mentre il mouse negli anni '80.



By Photographed by Andreas Franzkowiak (User:Bullenwächter) - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=61373639>

# IL WHIRLWIND, OVVERO L'ANTENATO DEI «PERSONAL COMPUTER»

Il Whirlwind è famoso per due motivi:

1. È il primo computer ad essere definito come «real-time», ovvero i cui tempi di elaborazione delle informazioni erano talmente rapidi da sembrare che una risposta ad un'interazione utente avvenisse «istantaneamente»
2. Era noleggiabile da chiunque per 15 minuti. Questo fece sì che anche i non-ricercatori potessero utilizzarlo e ciò contribuì alla nascita di un nuovo bisogno, ovvero quello di avere un computer personale per sé

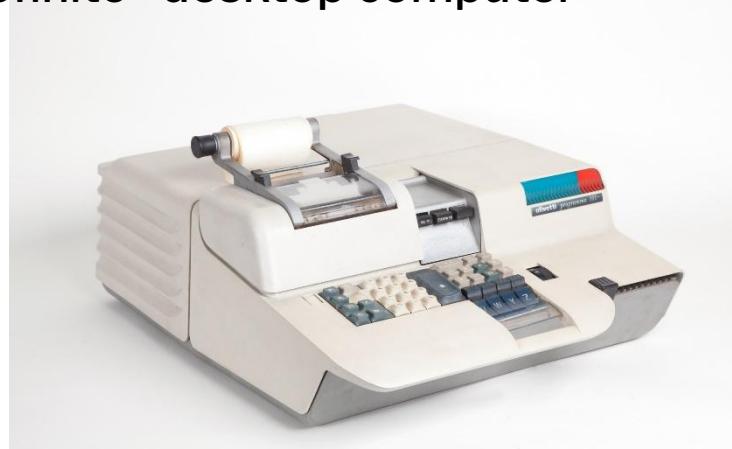
By Daderot - Own work, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=8894406>



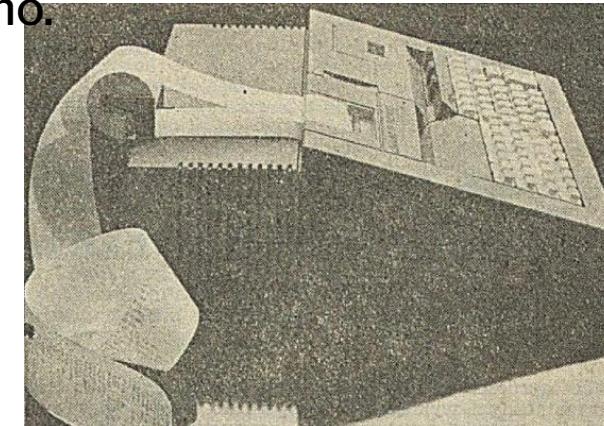
# I COMPUTER OLIVETTI

L'azienda italiana Olivetti occupa un posto importante nello sviluppo dei personal computer dagli anni '60.

Olivetti «Programma 101» è un calcolatore programmabile, uno dei primi al mondo ad essere definito «desktop computer»

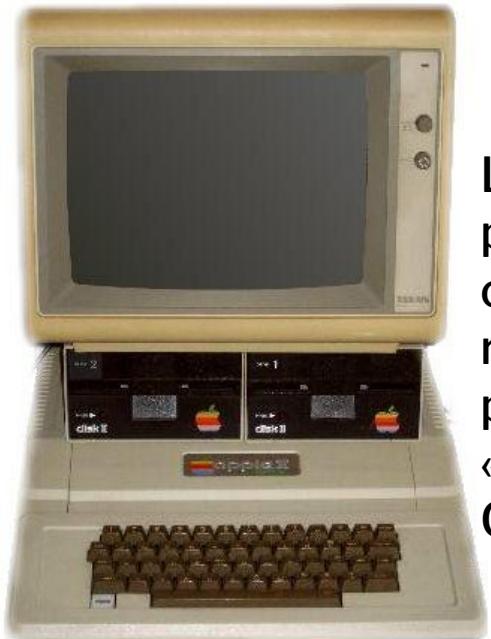
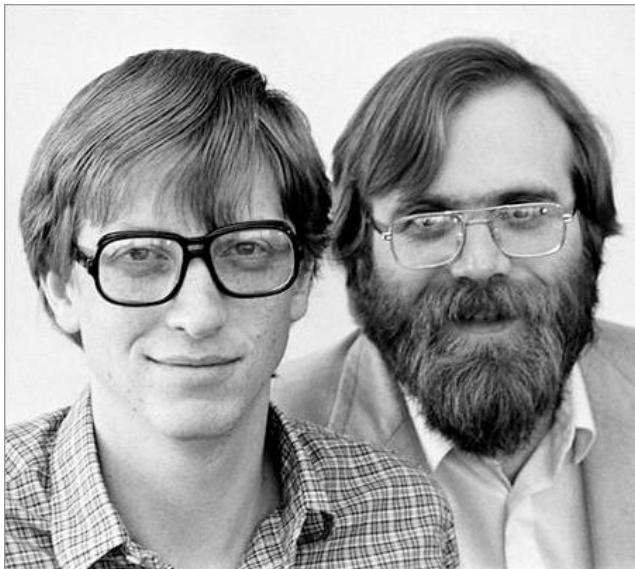


Olivetti «P6040» è un vero e proprio computer, il primo al mondo ad essere dotato di lettore floppy disk integrato, e dal peso e volume ridottissimo.



# MICROSOFT E APPLE

Nel 1975 Bill Gates e Paul Allen inventano un nuovo linguaggio di programmazione, BASIC, e decidono di commercializzarlo fondando una nuova azienda, Microsoft.



La Apple è la prima azienda a commercializzare i propri prodotti come «Personal Computer»

Nel 1976, Steve Jobs e Steve Wozniak, convinti che il computer sia prossimo a diventare un oggetto di consumo, fondano la Apple, con lo scopo di renderlo «al pari di un elettrodomestico».



# E POI?

Qui si conclude il nostro primo excursus nella storia dei computer.

D'ora in poi sarà necessario conoscere alcune nozioni software più importanti.

Continueremo con la narrazione dopo che avremo introdotto i **sistemi operativi**.

# LA PROSSIMA VOLTA...

Inizieremo a spostarci verso il *software*, andando a vedere

1. Come il computer *rappresenta* le informazioni (numeri, testo...)
2. Che cos'è un *programma*
3. Come il computer esegue le istruzioni contenute in un programma