Algoritmi e Codifica

Liceo Manzoni Lecco / Progetto Telerileviamo

Algoritmo: sequenza di operazioni

é la base della Programmazione: costruzione di SOFTWARE per la risoluzione di problemi, esecuzione di funzioni / azioni

creazione di Programmi / Applicazioni / App (software appunto)

Le macchine eseguono le azioni impostate nell'algoritmo, ricevendo dei dati dall'esterno (**INPUT**) e fornendo risultati in uscita (**OUTPUT**)

Possiamo schematizzare gli algoritmi con dei diagrammi FLOW-CHART

Istruzioni

La macchina è “stupida” ovvero necessita di “istruzioni” per svolgere le operazioni richieste. Le istruzioni possono essere di diverso tipo:

SEQUENZA: Istruzioni ripetute una dopo l’altra

ITERAZIONE: Ripetizione della stessa istruzione n volte (ciclo)

SELEZIONE: Istruzione che prevede una scelta

Variabili

Nella programmazione si utilizzano variabili come in matematica, corrispondenti a numeri o a caratteri (o a colori o a suoni). Queste informazioni sono CODIFICATE ovvero si trovano in formato numerico binario all'interno dei calcolatori, per essere poi DECODIFICATE dalle periferiche.

Ad ognuna di esse assegnamo un valore (Assegnazione) oppure lo possiamo ricevere dall'esterno (da un INPUT) e possiamo attraverso elaborazioni matematiche determinare dei risultati da comunicare all'esterno (attraverso un OUTPUT)

IL MODELLO DI VON NEUMANN

John Von Neumann

(1903-1957)

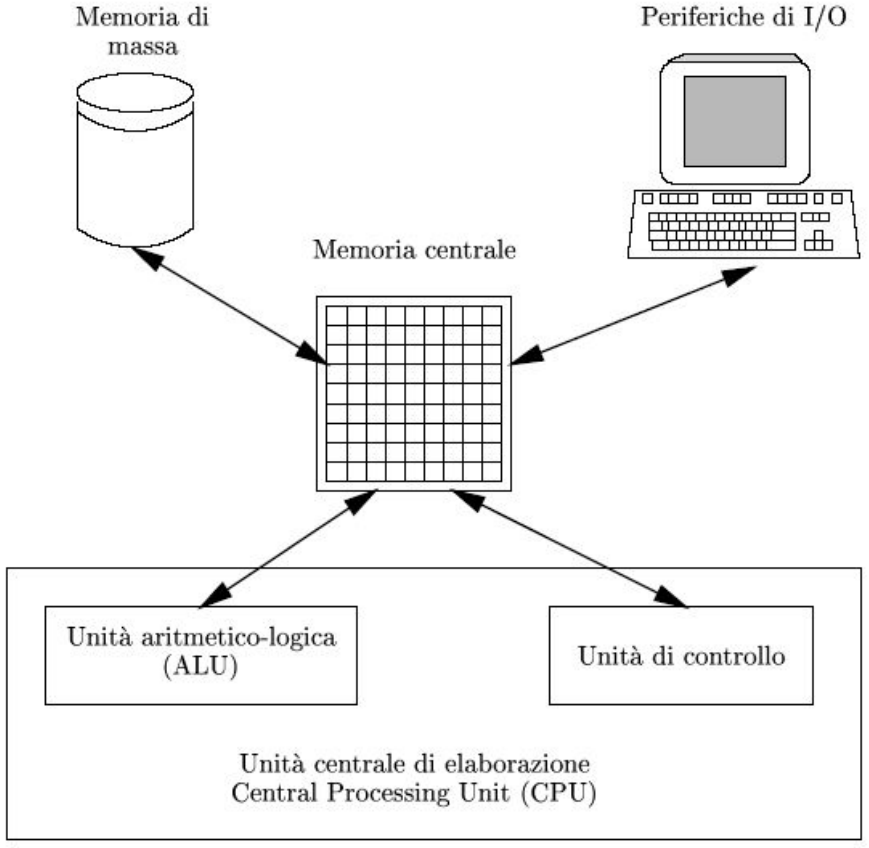
elabora l'architettura dei calcolatori moderni

CPU (UC+ALU)

Memoria centrale

(ROM+RAM)

Periferiche di INPUT/OUTPUT



Linguaggi di programmazione

La macchina elabora i dati attraverso una **CPU** ovvero CENTRAL PROCESSING UNIT (processore) utilizzando le varie memorie a disposizione (ad esempio la memoria RAM).

Tutti dati sono immagazzinati in forma di **bit** (cifre binarie) perciò programmare in questo modo sarebbe molto lento (tutto andrebbe scritto in **“LINGUAGGIO MACCHINA”** e il risultato all’occhio umano sarebbe fatto di stringhe come la seguente:0111001000101101O).

Vi sono dei **LINGUAGGI INTERMEDI** che riescono a tradurre semplici comandi in linguaggio macchina (ad esempio ASSEMBLER) e infine **LINGUAGGI DI ALTO LIVELLO** che riescono a tradurre delle scritture in codice in vere e proprie istruzioni da far eseguire alla macchina.

Linguaggi di programmazione

PASCAL, COBOL, FORTRAN, C, C++, JAVA, PYTHON, …

sono solo alcuni nomi dei **LINGUAGGI che traducono le istruzioni degli Algoritmi in veri e propri PROGRAMMI eseguibili dalla macchina**.

I programmi/applicazioni sono poi eseguiti tramite un software di sistema (SISTEMA OPERATIVO) che coordina e gestisce l'esecuzione di tutte le funzioni della macchina. Esempi di sistema operativo: Windows, Linux, IOS, Android,....

Esempio: Operazione tra matrici

Problema: data una matrice **nxm** (n righe ed m colonne) di numeri interi **a**, con **0<=a<=9** calcolare la matrice che contiene i numeri **9-n.**

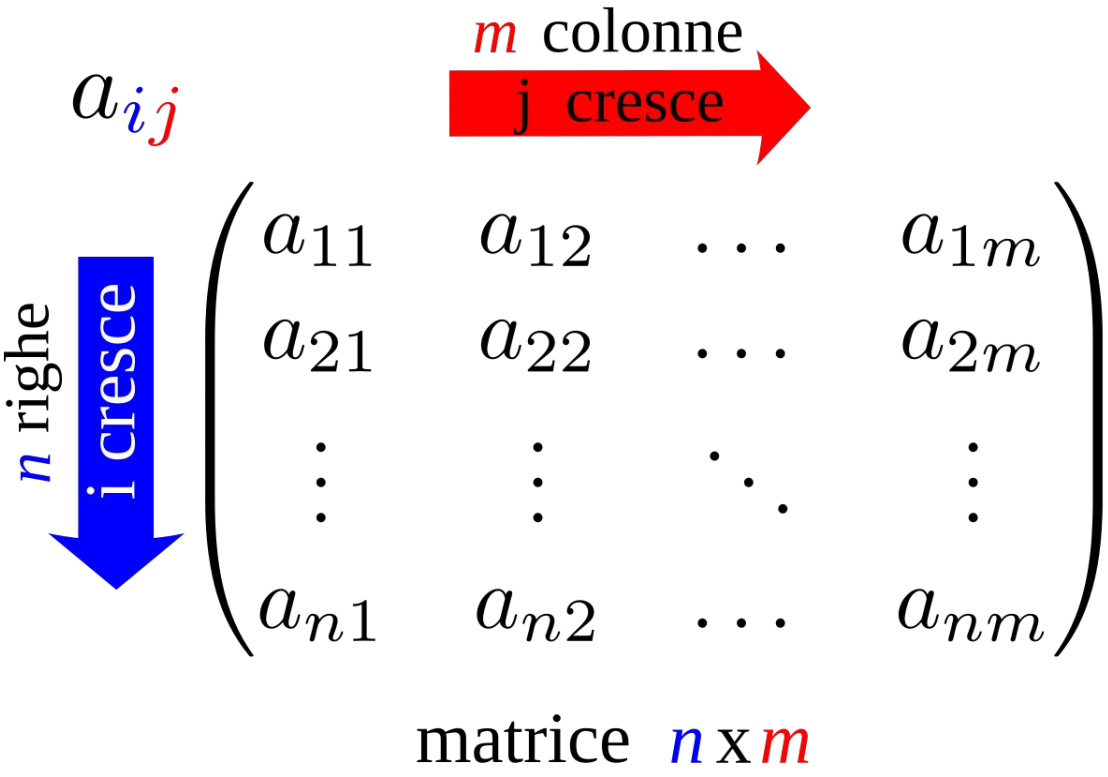
Per esempio partendo dalla matrice 3x5 (3 righe e 5 colonne)

1	2	6	6	8
0	7	9	7	9
2	6	6	4	1

il risultato dovrebbe essere la matrice

8	7	3	3	1
9	2	0	2	0
7	3	3	5	8

MATRICE NxM


a_{ij}
m colonne
j cresce
n righe
i cresce
$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$
matrice n x m

Quali INPUT???

Dobbiamo poter inserire i dati relativi alla Matrice (i singoli numeri)

Tramite un'interfaccia (tastiera, da file,touchscreen…) dobbiamo comunicare alla macchina che i dati da elaborare sono: 1, 2, 6, 6, 6, 8, 0, 7,...etc

Dove finiscono i nostri **INPUT**? è necessario che ci sia uno spazio di memoria allocato appositamente per questo tipo di dato. In questo caso si tratta di 9 numeri interi (INTEGER)che dovranno avere ciascuno un relativo spazio di memoria dedicato.

Il programmatore deve quindi "creare" delle variabili che possano contenere questi numeri

Come devo elaborare ciascun dato?

Su ciascuno dei 9 numeri devo eseguire delle operazioni per giungere al risultato finale. L'espressione analitica della funzione in questione potrebbe essere:

f(x)=9-x

Affinchè l'operazione sia possibile è necessario che si inserisca un numero compreso tra 0 e 9, per questo dovremo operare un CONTROLLO sui dati inseriti con un meccanismo di **SELEZIONE**, che mi permette di inserire in memoria solo dati corretti e richiedere all'utente un nuovo inserimento in caso di dato in input che non risponda agli standard richiesti.

Dominio e Codominio di questa funzione sono: {0,1,2,3,4,5,6,7,8,9}

Devo ripetere più volte la stessa operazione o…

Questa operazione va svolta più volte (9 volte!) ma è evidente che la funzione è sempre la stessa, cambiano solo i valori che di volta in volta le sottopongo.

Posso scrivere una sola volta questa operazione inserendola in un meccanismo di **ITERAZIONE** detto anche **CICLO**

Se la matrice fosse 10 X 10 avremmo 100 numeri da inserire e da processare, perciò è evidente che scrivere 100 volta la stessa istruzione sarebbe dispendioso, mentre scrivendola una sola volta all'interno di un meccanismo ciclico, la macchina riproporrà la richiesta di input e l'elaborazione più volte, il numero necessario a coprire tutti i dati in ingresso.

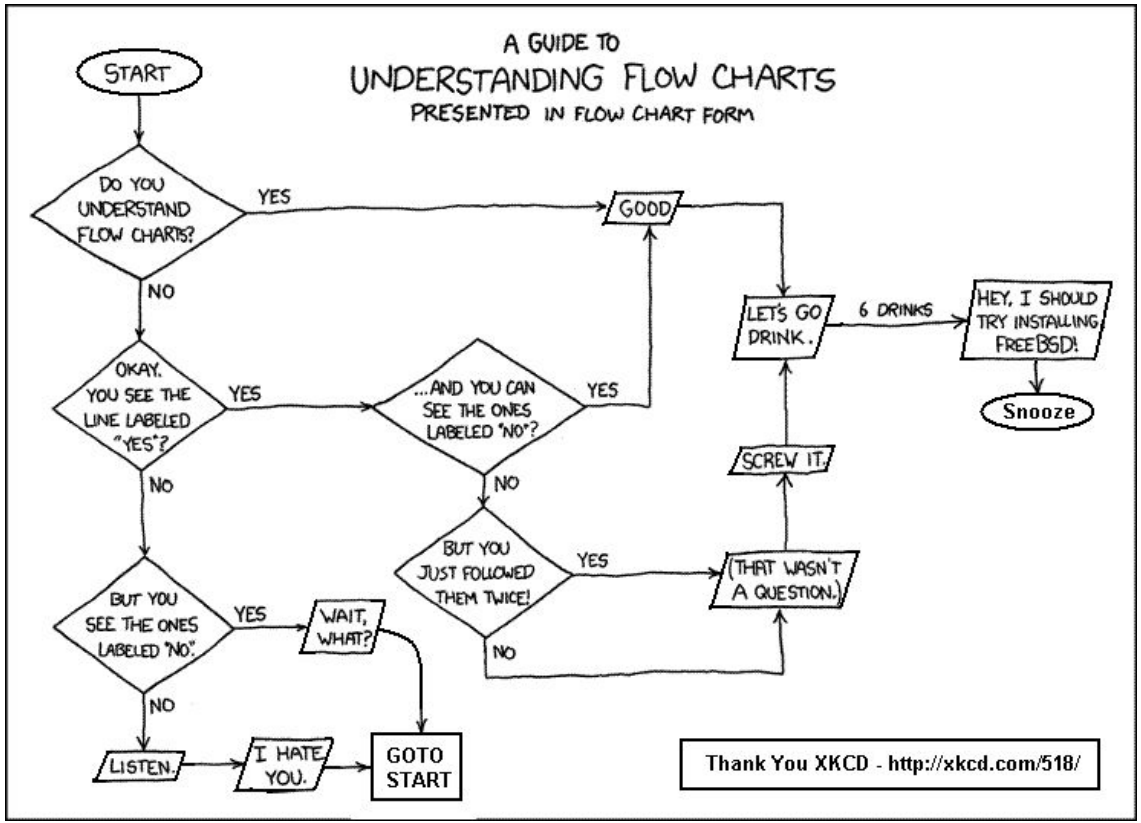
Quale Output?

Una volta che i dati sono stati correttamente inseriti e le elaborazioni sono state effettuate, quali saranno gli **OUTPUT** e come si ottengono?

Devo ordinare alla macchina di eseguire delle operazioni di comunicazione con periferiche di output (schermo, stampante, suoni, files, etc) che mi permettano di acquisire i risultati. Per questo specifico problema dovrò avere un ciclo di restituzione che mi “stampi” i numeri desiderati.

Flow CHART: schema a blocchi/Diagramma di flusso

Per esemplificare la catena logica di azioni che dovranno tradursi in ISTRUZIONI che dobbiamo impartire alla macchina utilizziamo blocchi fatti in questo modo:



START/END

inizio/fine: sono le operazioni che mi permettono di attivare o disattivare l’esecuzione del programma

INPUT/OUTPUT

Trasferimento di informazioni: lettura dati in ingresso, scrittura dati in uscita, visualizzazione dati intermedi

in generale COMUNICAZIONE

Esecuzione

esecuzione di calcoli/elaborazione: sono le operazioni di calcolo o elaborazione dei dati, le “Azioni”

SELEZIONE

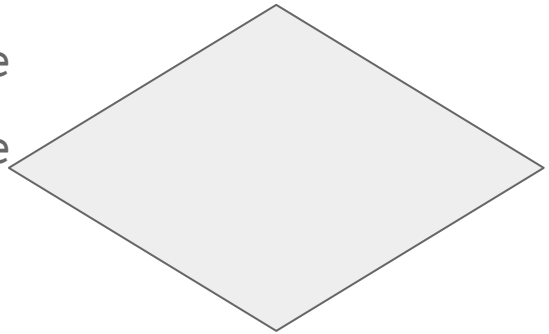
Assunzione di decisioni/selezione: ogni volta che si deve attuare una scelta in base alla quale si possono svolgere operazioni oppure no, prendere “vie diverse” per raggiungere risultati condizionati alla nostra scelta.

La condizione può avere valore logico

VERO o FALSO, perciò ci sono due diverse

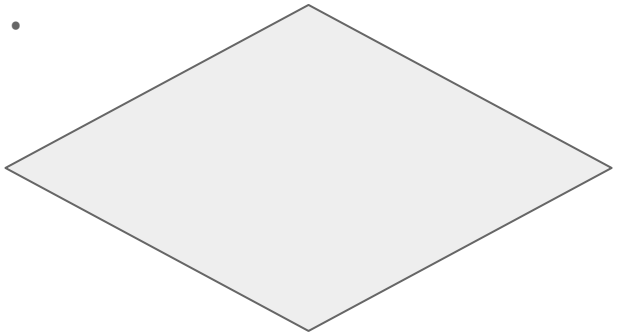
uscite e quindi due diverse “strade” che

il programma può seguire.



ITERAZIONI

Esecuzione di iterazioni: ripetizione di sequenze di operazioni. E’ governato da alcune condizioni, quando il valore è VERO si procede all’iterazione successiva, quando il valore è FALSO si esce dal ciclo e si prosegue con il programma.

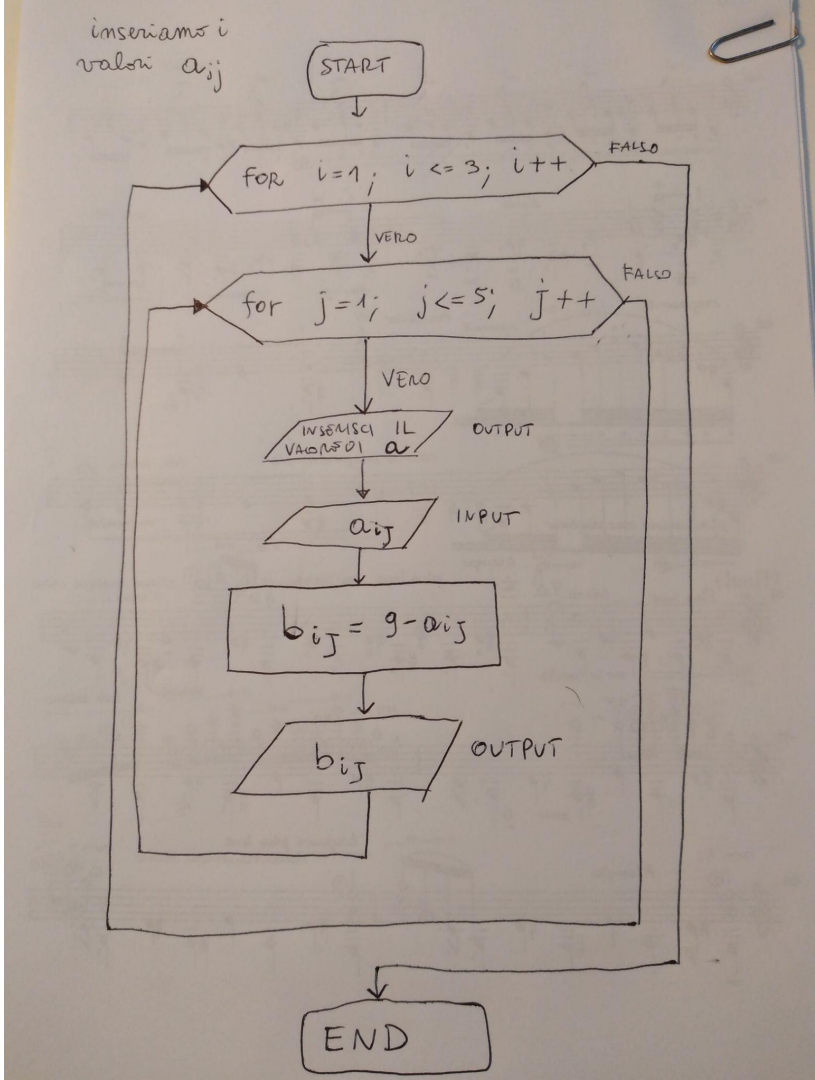


Inserimento dei valori

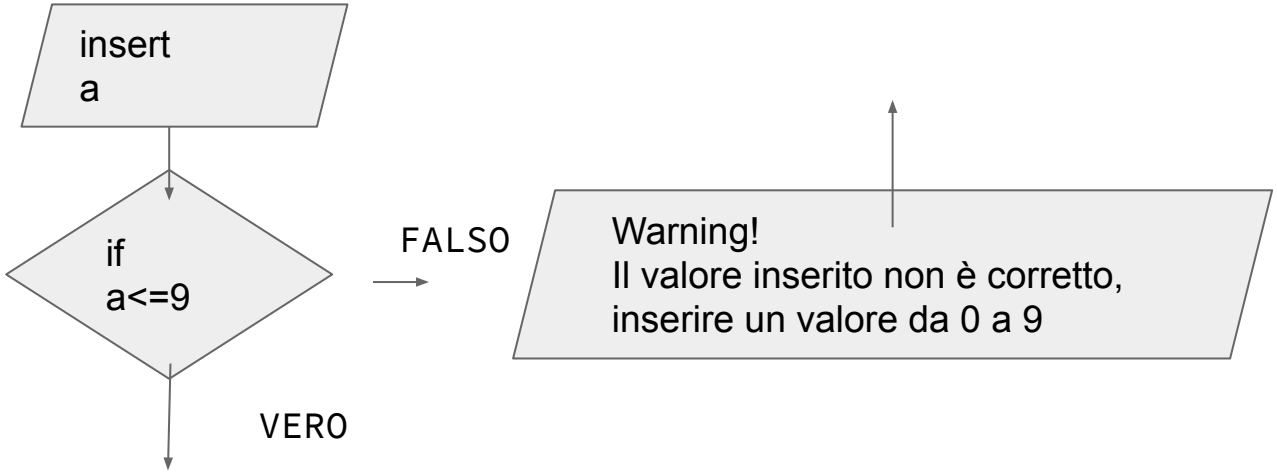
Per acquisire i valori a_{ij}(input) utilizziamo due cicli annidiati, eseguiamo il calcolo

$b_{ij}=9-a_{ij}$

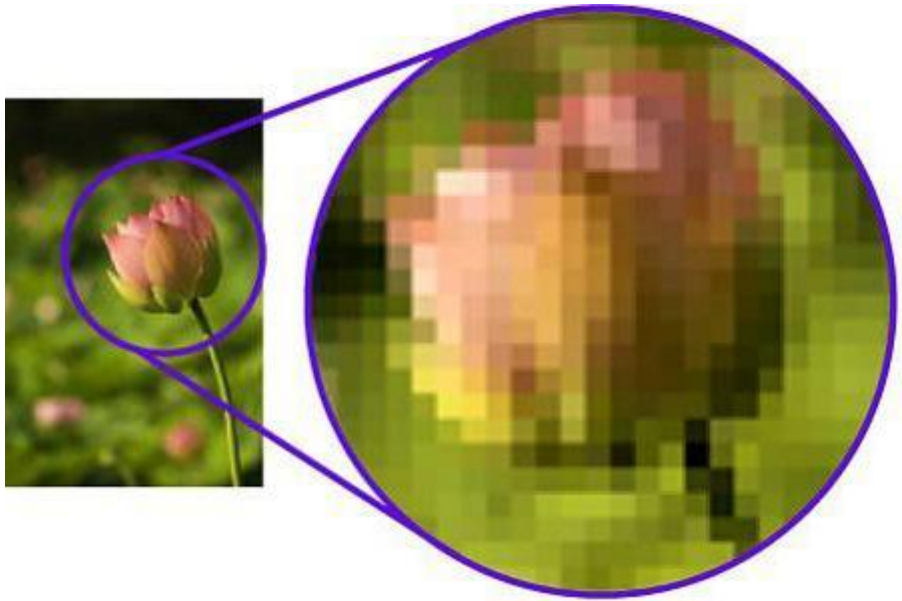
e infine l'output costituito dai valori b_{ij}



Controlli / Warning



Matrici di PIXEL: le immagini digitali

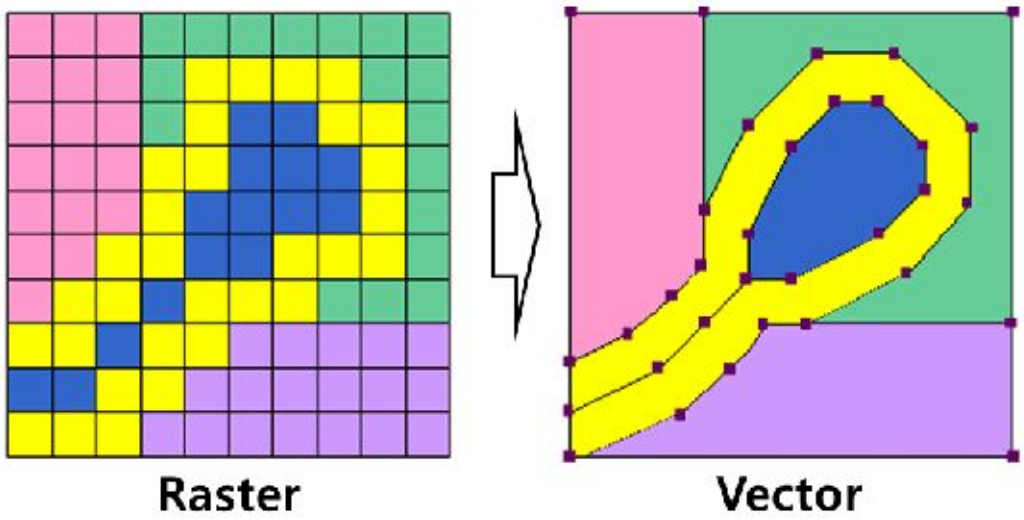


Le operazioni svolte sugli elementi di una matrice possono corrispondere a operazioni svolte su elementi di immagine: ogni dato, ogni informazione è tradotta in numero all’interno di un calcolatore, perciò possiamo pensare ad una immagine digitale come a una matrice di pixel

Un esempio: operare su immagini raster

Una immagine RASTER non è altro che una matrice di PIXEL (picture elements) in cui ogni elemento è caratterizzato da un valore numerico che codifica un colore, quello che noi possiamo visualizzare tramite schermo o stampa.

La manipolazione di immagini raster a livello di algoritmo è un insieme di operazioni da calcolare sugli elementi di questa matrice



Caratteristiche di un RASTER

1. Dimensioni (in peso) - il numero totale di pixel nell'immagine è calcolato in CB (MB, GB). Il quadro più grande e più complesso, tanto più pesa.
2. Risoluzione - il numero di pixel per pollice (ppi) nelle linea immagini, foto o punti per pollice (dpi) in illustrazioni stampate. Più grande è il valore, migliore è la foto, più chiaro. immagini a risoluzione standard Internet - 72 ppi, layout di stampa - 300 ppi.
3. Modulo colore definisce le tinte base. Questo può essere comune RGB, quando il rosso, verde e blu sono presenti in una data quantità di ciascun pixel e miscelazione per formare il colore desiderato. Per preparare i layout spesso usano CMYK - modulo composto da ciano, magenta, giallo e nero. LAB - è "brillante", il rosso-verde e blu-giallo; Scala di grigi - sfumature di grigio.
4. Quanti bit sono codificati in ogni pixel dipende dal colore dell'immagine. Nelle immagini monocromatiche ogni punto pesa 1 bit. Se il modello di pixel 4 bit consiste di 16 colori. 8 BPP dare 256 colori, 16 bit - 65K colori 24 bit -. 16 milioni di colori ..

CODIFICA

A questo punto bisogna capire come può un’immagine, un suono, un testo essere elaborato in un calcolatore che normalmente è una macchina che lavora su impulsi elettrici o valori di campo magnetico, perciò con dei valori numerici, ovvero NUMERI.

Le informazioni sono CODIFICATE per essere sottoposte alle elaborazioni contenute nei software che eseguono gli algoritmi impostati dai programmatori

Il linguaggio Macchina

Come abbiamo visto ogni INFORMAZIONE viene immagazzinata, riconosciuta e gestita in un linguaggio che la macchina possa gestire. Quale linguaggio?

Le macchine riescono a gestire e memorizzare i dati tramite il sistema di numerazione binario, cioè quello formato da due sole cifre: 0 e 1

Il motivo è il fatto che che i dispositivi fisici sono formati da celle di memoria che si comportano come “interruttori”, sono cioè dispositivi che hanno 2 possibili stati

0 corrisponde ad uno stato (spento, campo elettrico -, un certo valore del campo magnetico)

1 corrisponde ad uno stato diverso (acceso, campo elettrico +, diverso valore del campo magnetico)

Binary Digit: il Bit e il Byte

Ogni unità singola di memoria contiene una cifra che può essere 0 o 1, perciò questa cifra (DIGIT) binaria (BINARY) è chiamata BIT.

Per immagazzinare le informazioni servono molte di queste unità di memoria, che sono organizzate in gruppi di 8 BIT che si chiamano BYTE

Esempio: 01101100 è formato da 8 cifre, forma un BYTE di memoria. Quanti numeri (o diverse informazioni) possiamo immagazzinare in un BYTE?

Con 2 bit 2^2 Valori

Con 2 bit:

00	0
01	1
10	2
11	3

Con 3 bit 2^3 valori

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

Con 8 bit=1Byte avremo 2^8 valori (256)

00000000	0	00001010	10
00000001	1	10001011	11
00000010	2	etc.	etc.
00000011	3	……..	….
00000100	4	……..	…..
00000101	5	11111011	251
00000110	6	11111100	252
00000111	7	11111101	253
00001000	8	11111110	254
00001001	9	11111111	255

Determinare il numero decimale corrispondente

Il sistema binario è posizionale come il sistema decimale, possiamo trasformare un numero facilmente ricordando che ad ogni posizione corrisponde una potenza di 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	**1**	**0**	**0**	**1**	**1**	**0**	**1**

Il numero in tabella è 0*128+1*64+0*32+0*16+1*8+1*4+0*2+1*1

cioè 77

Come immagazzinare le informazioni

Ora sappiamo che in un BYTE possiamo immagazzinare 256 valori diversi di un dato, ma questo non deve essere necessariamente un numero. Può essere

UN CARATTERE DI TESTO

UN COLORE

UN SUONO

UN'ISTRUZIONE

LA CODIFICA

Perchè uno o più byte possano essere visualizzati (colori) oppure sentiti (suoni) oppure eseguiti (ad esempio l'invio di una stampa) occorre che il contenuto del byte venga elaborato dalla CPU e inviato alle periferiche (schermo, uscita audio, porta usb, etc…) che sono in grado di ricevere e DECODIFICARE l'informazione contenuta nel BYTE

Esistono quindi degli STANDARD con i quali vengono codificati i dati di vario tipo

Il sistema esadecimale

Prima di passare a esaminare alcune codifiche è utile sapere che esiste una rappresentazione tramite sistema ESADECIMALE
Si può passare da un byte scritto in binario (8 cifre, ciascuna con 2 valori) a un byte scritto in esadecimale (2 cifre, ciascuna con 16 valori)

Infatti $2^8 = 16^2$

Le cifre sono:

0 1 2 3 4 5 6 7 8 9 A B C D E F

4 BIT= 1 CIFRA ESADECIMALE

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Come rappresentare i byte in esadecimale

Perciò il byte 01100100 può essere rappresentato in esadecimale così:

0110 = 6 e 0100 = 4 quindi

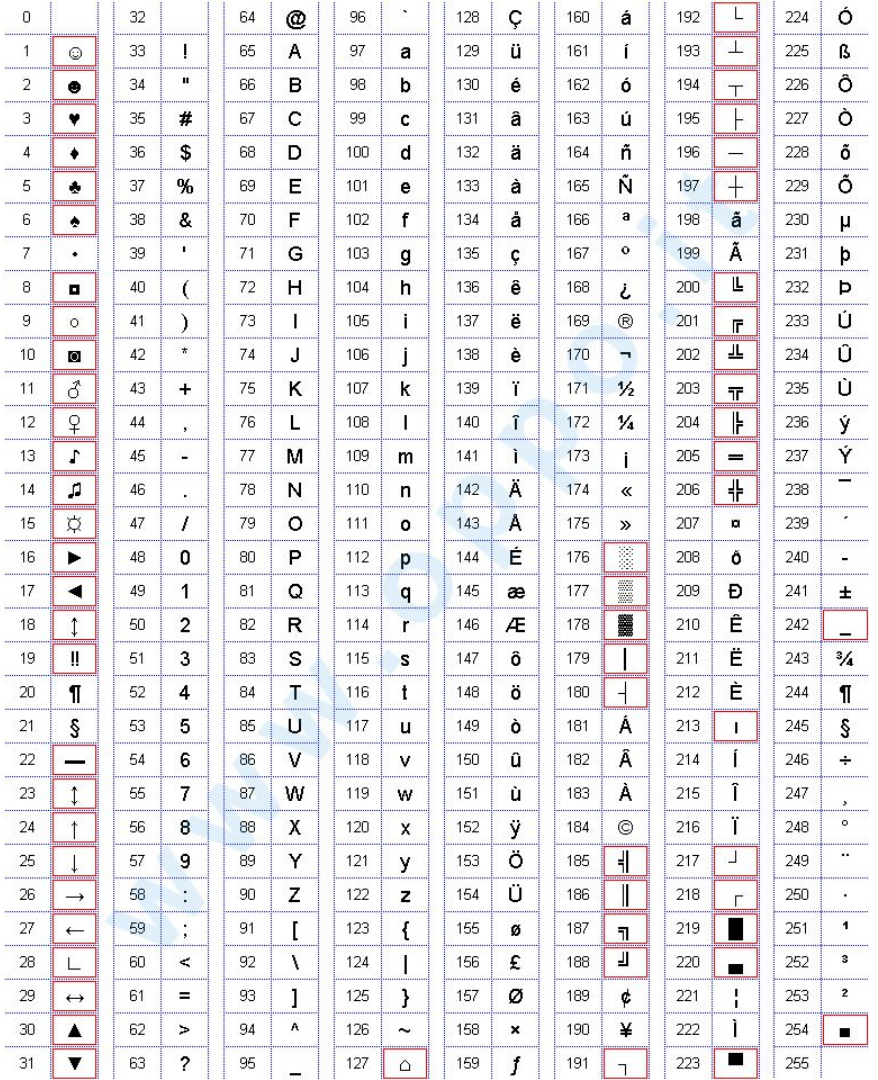
01100100=64

altro esempio:

10110001 = B1

La codifica dei Caratteri

La tabella a 8 bit utilizzata per la codifica dei caratteri si chiama ASCII e comprende 256 diversi caratteri, tuttavia oggi si utilizza la codifica UNICODE che può essere da 16 bit a 21 bit (2^{16} -2^{21} caratteri)



La codifica dei numeri

Per poter utilizzare numeri nel campo Reale, e non solo numeri interi da 0 a 255 si sono elaborati altri sistemi di codifica. Il più usato per dati scientifici è il numero a virgola mobile (FLOATING POINT)

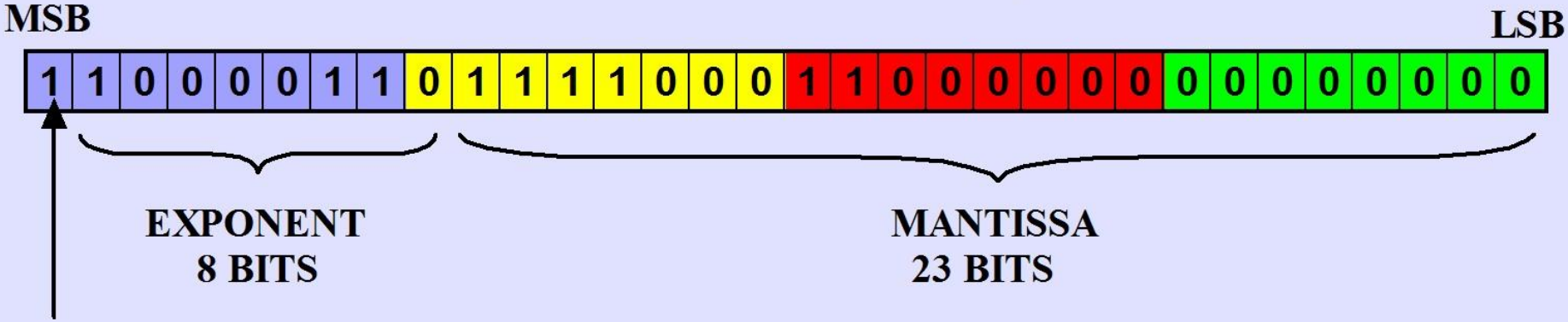
Di solito per ogni numero abbiamo 32 bit (o anche 64 bit per maggiore precisione), ed è scritto in forma esponenziale:

1bit per il segno

8 bit per l'esponente (si intende l'esponente relativo ad una potenza di base 10)

24 bit per la mantissa (parte decimale)

FLOATING POINT FORMAT IEEE-754, 32 BITS



EXAMPLE: -248.75

HEXADECIMAL: C3 78 C0 00

I colori

Il colore di una unità grafica (PIXEL) possono avere diverse codifiche:

1bit: B/N (Bianco e nero)

1Byte=8bit Scala di grigi (256 sfumature di grigio)

Il colore RGB permette di avere circa 16 milioni di colori utilizzando 24 bit per pixel

8bit: red

8bit: green

8bit: blue

Ogni colore si ottiene con la SINTESI ADDITIVA dei 3 colori (2^{24} colori)

Sintesi additiva(schermi) e sottrattiva (stampe)

	R	G	B	值
黑色	0	0	0	#000000
象牙黑	41	36	33	#292421
灰色	192	192	192	#C0C0C0
冷灰	128	138	135	#808A87
石板灰	112	128	105	#708069
暖灰色	128	128	105	#808069
白色	255	255	255	#FFFFFF
古董白	250	235	215	#FAEBD7
天蓝色	240	255	255	#F0FFFF
白烟	245	245	245	#F5F5F5
白杏仁	255	235	205	#FFFFCD
cornsilk	255	248	220	#FFF8DC
蛋壳色	252	230	201	#FCE6C9
花白	255	250	240	#FFFAF0
gainsboro	220	220	220	#DCDCDC
ghostWhite	248	248	255	#F8F8FF
蜜露橙	240	255	240	#F0FFF0
象牙白	250	255	240	#FAFFF0
亚麻色	250	240	230	#FAF0E6
navajoWhite	255	222	173	#FFDEAD
old lace	253	245	230	#FDF5E6
海贝壳色	255	245	238	#FFF5EE
雪白	255	250	250	#FFFAFA
红色	255	0	0	#FF0000
砖红	156	102	31	#9C661F
镉红	227	23	13	#E3170D
珊瑚色	255	127	80	#FF7F50
耐火砖红	178	34	34	#B22222
印度红	176	23	31	#B0171F
栗色	176	48	96	#B03060
粉红	255	192	203	#FFC0CB
草莓色	135	38	87	#872657
橙红色	250	128	114	#FA8072
番茄红	255	99	71	#FF6347
桔红	255	69	0	#FF4500
深红色	255	0	255	#FF00FF

	R	G	B	值
黄色	255	255	0	#FFFF00
香蕉色	227	207	87	#E3CF57
镉黄	255	153	18	#FF9912
dougello	235	142	85	#EB8E55
forum gold	255	227	132	#FFE384
金黄色	255	215	0	#FFD700
黄花色	218	165	105	#DAA569
瓜色	227	168	105	#E3A869
橙色	255	97	0	#FF6100
镉橙	255	97	3	#FF6103
胡萝卜色	237	145	33	#ED9121
桔黄	255	128	0	#FF8000
淡黄色	245	222	179	#F5DEB3
棕色	128	42	42	#802A2A
米色	163	148	128	#A39480
锻浓黄土色	138	54	15	#8A360F
锻棕土色	135	51	36	#873324
巧克力色	210	105	30	#D2691E
肉色	255	125	64	#FF7D40
黄褐色	240	230	140	#F0E68C
玫瑰紅	188	143	143	#BC8F8F
肖贡土色	199	97	20	#C76114
标土棕	115	74	18	#734A12
乌贼墨棕	94	38	18	#5E2612
赫色	160	82	45	#A0522D
马棕色	139	69	19	#8B4513
沙棕色	244	164	96	#F4A460
棕褐色	210	180	140	#D2B48C
蓝色	0	0	255	#0000FF
钴色	61	89	171	#3D59AB
dodger blue	30	144	255	#1E90FF
jackie blue	11	23	70	#0B1746
锰蓝	3	168	158	#03A89E
深蓝色	25	25	112	#191970
孔雀蓝	51	161	201	#33A1C9
土耳其玉色	0	199	140	#00C78C

	R	G	B	值
浅灰蓝色	176	224	230	#B0E0E6
品蓝	65	105	225	#4169E1
石板蓝	106	90	205	#6A5ACD
天蓝	135	206	235	#87CEEB
青色	0	255	255	#00FFFF
绿土	56	94	15	#385E0F
靛青	8	46	84	#082E54
碧绿色	127	255	212	#7FFFD4
青绿色	64	224	208	#40E0D0
绿色	0	255	0	#00FF00
黄绿色	127	255	0	#7FFF00
钴绿色	61	145	64	#3D9140
翠绿色	0	201	87	#00C957
森林绿	34	139	34	#228B22
草地绿	124	252	0	#7CFC00
酸橙绿	50	205	50	#32CD32
薄荷色	189	252	201	#BDFCC9
草绿色	107	142	35	#6B8E23
暗绿色	48	128	20	#308014
海绿色	46	139	87	#2E8B57
嫩绿色	0	255	127	#00FF7F
紫色	160	32	240	#A020F0
紫罗蓝色	138	43	226	#8A2BE2
jasoa	160	102	211	#A066D3
湖紫色	153	51	250	#9933FA
淡紫色	218	112	214	#DA70D6
梅红色	221	160	221	#DDA0DD

	<<< Control 1 - Adjust Hue(Color) >>>	<<<= Control 2 - Adjust FROM Colour to White =>>>																	
		1			2			3			4			5			6		
		R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
Red	23	255	0	0	255	100	100	255	150	150	255	200	200	255	230	230	255	255	255
	22	255	100	0	255	110	20	255	120	100	255	165	155	255	230	210	255	255	255
Orange	21	255	150	0	255	140	20	255	180	40	255	220	120	255	240	190	255	255	255
	20	255	200	0	255	210	20	255	230	100	255	240	150	255	255	200	255	255	255
Yellow	19	255	255	0	255	255	20	255	255	80	255	255	150	255	255	210	255	255	255
	18	200	255	0	200	255	20	220	255	80	230	255	150	235	255	180	255	255	255
Chartreuse	17	150	255	0	150	255	20	200	255	40	220	255	150	230	255	170	255	255	255
	16	100	255	0	110	255	20	150	255	40	200	255	150	230	255	160	255	255	255
Green	15	0	255	0	100	255	100	150	255	150	200	255	200	225	255	225	255	255	255
	14	0	255	100	20	255	150	100	255	180	180	255	210	235	255	220	255	255	255
Aquamarine	13	0	255	150	20	255	160	100	255	200	160	255	210	200	255	220	255	255	255
	12	0	255	200	20	255	200	40	255	220	140	255	230	200	255	255	255	255	255
Cyan	11	0	255	255	20	255	255	40	255	255	150	255	255	180	255	255	255	255	255
	10	0	200	255	20	210	255	40	230	255	60	240	255	170	255	255	255	255	255
Azure	9	0	150	255	20	180	255	40	200	255	60	230	255	150	255	255	255	255	255
	8	0	100	255	20	150	255	40	180	255	60	200	255	140	255	255	255	255	255
Blue	7	0	0	255	20	100	255	40	150	255	80	180	255	160	220	255	255	255	255
	6	100	0	255	120	20	255	140	100	255	150	150	255	200	200	255	255	255	255
Violet	5	150	0	255	150	50	255	180	100	255	220	140	255	230	180	255	255	255	255
	4	200	0	255	200	20	255	220	60	255	240	90	255	255	160	255	255	255	255
Magenta	3	255	0	255	255	60	255	255	100	240	255	150	250	255	180	255	255	255	255
	2	255	0	200	255	20	220	255	40	230	255	140	240	255	170	255	255	255	255
Rose	1	255	0	150	255	20	200	255	40	200	255	100	200	255	160	240	255	255	255
	0	255	0	100	255	20	120	255	80	140	255	100	160	255	140	180	255	255	255

Operare sui raster

Torniamo alle nostre immagini.

Quali operazioni possiamo compiere sulle nostre “matrici” di pixel?

Esempio in scala di grigi

In una immagine a scala di grigi il pixel occupa 1BYTE (8bit)

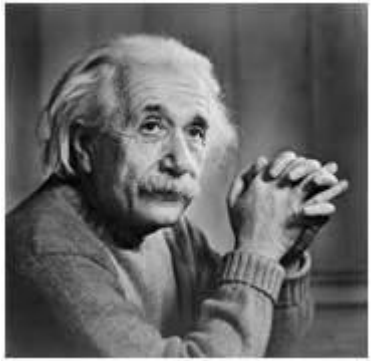
perciò i colori vanno da 0 (BLACK) a 255 (WHITE)

Cosa succede se applico a ciascun pixel della matrice BITMAP la funzione

f(x)=255-x

????

INVERT COLOR



INPUT x

f(x)=255-x

OUTPUT f(x)

Altro esempio (logaritmico)

se la funzione fosse invece

f(x)=clog(x+1)

alcuni valori:

f(0)= 0

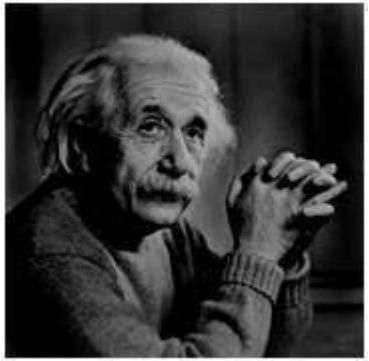
f(1)= clog(2)

f(2)= clog(3)

…

f(255)=clog(256)

Log transformation

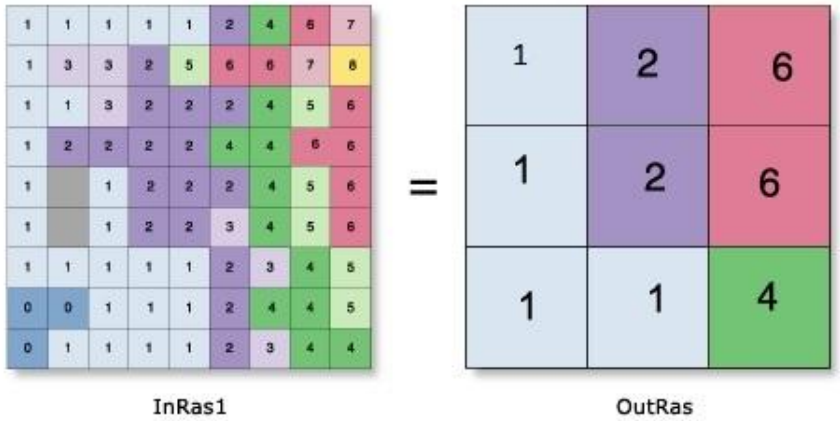


INPUT x

f(x)=clog(x+1)

OUTPUT f(x)

Un esempio: aggregazione al valore più frequente



Un ulteriore tipo di elaborazione può essere dato da funzioni che non operano su un singolo pixel ma su un gruppo di pixel, per esempio per rendere omogenee delle aree assegnando un valore statisticamente rilevante (media, moda etc)