# 1.1 Fondamenti

## Algoritmi

Prima di vedere nel dettaglio il funzionamento di un *computer* (argomento che verrà approfondito nel prossimo capitolo; per ora può essere considerato come un elaboratore che esegue istruzioni) è necessario farsi un'idea del modo in cui esso interpreta i problemi.

Un computer non ha una propria capacità di "elaborazione creativa", cioè non è in grado di eseguire alcuna operazione se non viene opportunamente istruito. Spetta all'utente tradurre il problema in termini formali, individuare dati e incognite (gli elementi non noti, da determinare), schematizzare tutti i passaggi, prevedere tutti i possibili casi che si possono presentare e indicare opportunamente la via da seguire, in modo che la macchina possa arrivare alla soluzione. L'individuazione di una sequenza ordinata di istruzioni che porta alla risoluzione di un problema viene definita **algoritmo**. L'algoritmo, quindi, è una sequenza di operazioni e passaggi da compiere per risolvere un determinato problema. Le operazioni così definite, devono poi essere tradotte in opportuni linguaggi, in modo che le istruzioni possano essere "comprese" dal computer.

Nella vita di tutti i giorni, inconsapevolmente, si seguono degli algoritmi: per esempio, per preparare un piatto particolare si segue una ricetta, o per montare un mobile si svolgono determinate operazioni in sequenza, per sommare due numeri o per effettuare l'iscrizione a un corso. Nelle prossime sezioni verranno presentati due algoritmi: uno per risolvere un problema di tipo culinario (preparare una camomilla) e uno per risolvere un problema di tipo finanziario (acquistare dei capi d'abbigliamento ).

**Le istruzioni di un algoritmo**

Le istruzioni di un algoritmo devono essere:

- *interpretabili in un unico modo*, non possono essere ambigue;
- *elementari*, quelle più complesse devono essere suddivise in istruzioni più semplici, in modo che possano essere "capite" più facilmente dal computer;
- *realizzabili*, cioè devono poter essere eseguibili materialmente;

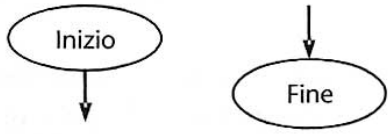
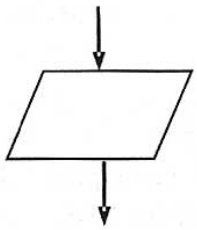
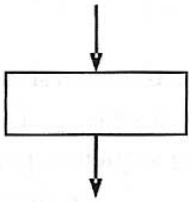
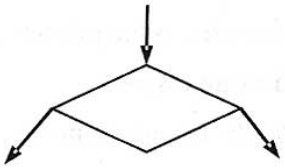
➢ *in numero finito*, se così non fosse, come potrebbe un computer arrivare a una conclusione? Lavorerebbe all'infinito senza poter restituire alcun risultato.

## Diagramma di flusso

L'algoritmo può essere rappresentato in vari modi, grafici o testuali. Uno dei metodi grafici più usati e conosciuti è il cosiddetto **diagramma di flusso**, ciascun componente del quale ha un significato ben determinato. Nella Tabella 1.1 sono riportati i vari elementi che possono costituire un diagramma di flusso e il relativo significato.

È importante sottolineare la differenza tra le proprietà degli elementi di formati diversi. Per esempio l'elemento a forma di rombo costituisce sempre una "diramazione": in base alla valutazione della condizione racchiusa, viene seguito un certo percorso invece di un altro.

**Tabella 1.1 Simboli che possono essere usati per la creazione del diagramma di flusso**

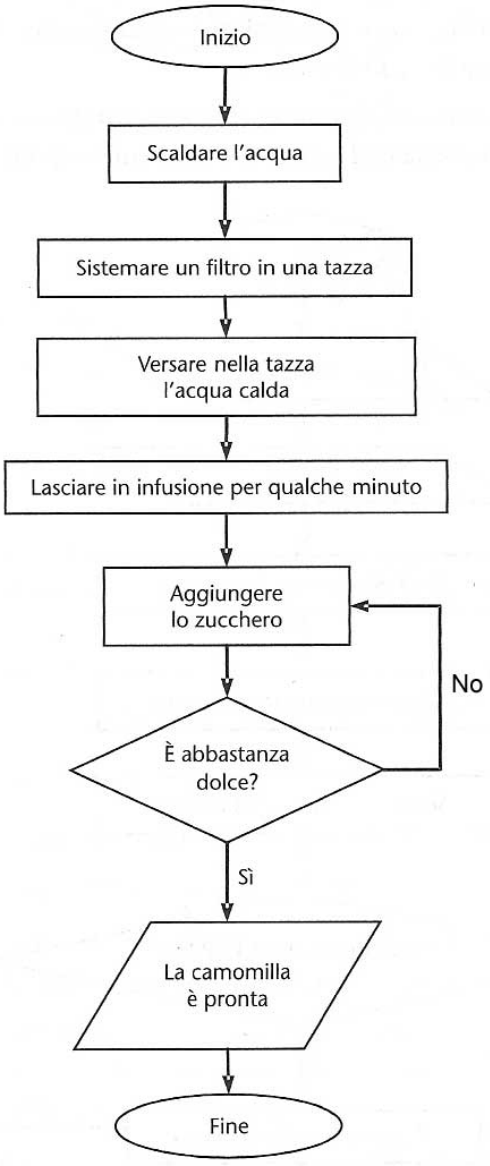
| Simbolo | Significato |
|---|---|
|   | **Inizio** e **fine** della sequenza di istruzioni |
|  | **Inserimento** ed **emissione** dei dati |
|  | **Istruzione** da eseguire |
|  | **Istruzione** che implica una scelta tra due possibili percorsi a seconda della valutazione di una certa condizione. |

## Progettare algoritmi non numerici

Si provi a pensare alle operazioni da seguire per preparare una camomilla:

- scaldare l'acqua,
- sistemare un filtro in una tazza,
- versare nella tazza l'acqua calda,
- lasciare in infusione per qualche minuto,
- aggiungere zucchero.

La sequenza di passi appena descritta costituisce un algoritmo molto semplice di tipo non numerico. Ecco una sua possibile rappresentazione grafica tramite diagramma di flusso.



## Progettare algoritmi numerici

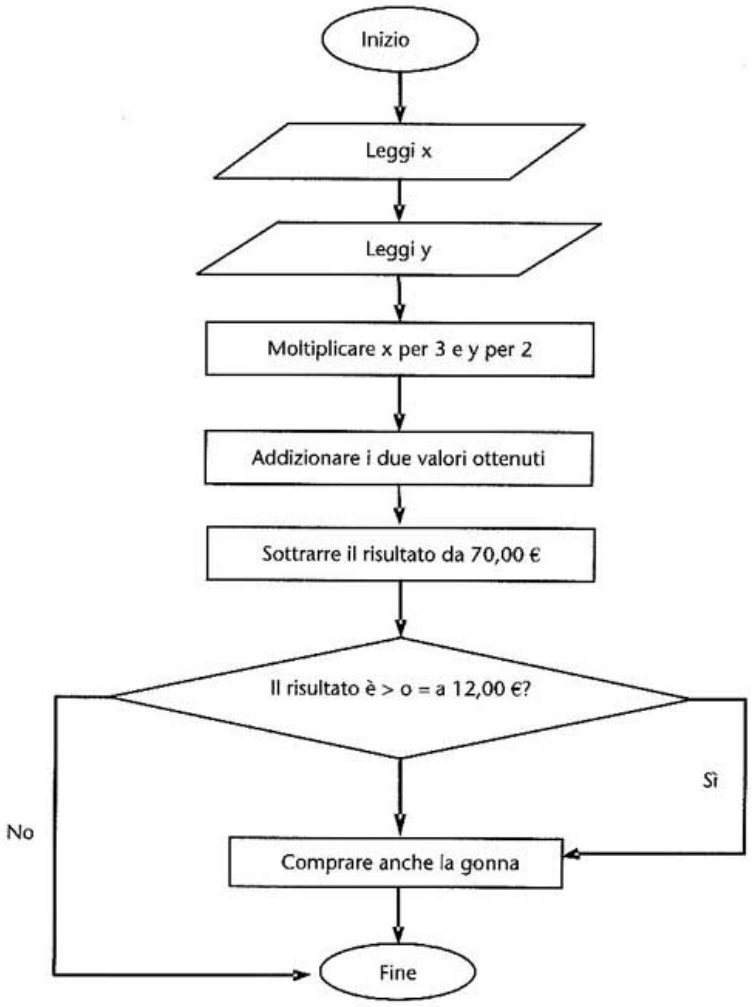
Si provi a pensare invece al seguente problema, risolvibile con l'applicazione di un algoritmo di tipo numerico:

> Si devono acquistare tre maglie che hanno un costo x e due camicie di costo y. Se si hanno a disposizione 70,00 €, rimangono i soldi per comprare anche una gonna da 12,00 €?

Si può definire il seguente algoritmo:

- moltiplicare per tre il costo di una maglia;
- moltiplicare per due il costo di una camicia;
- addizionare i due valori ottenuti (totale della spesa per le maglie e per le camicie);
- confrontare la differenza tra quanto si ha a disposizione (70,00 €) e la somma dei due valori: se è maggiore o uguale al costo della gonna (12,00 €), si può procedere anche al suo acquisto, altrimenti no.

Quello appena descritto è l'algoritmo espresso in termini informali. Ecco una sua possibile rappresentazione grafica tramite diagramma di flusso.



# Rappresentazione dei dati

Spesso si devono risolvere dei problemi matematici più o meno complessi: per esempio dopo aver fatto la spesa e aver pagato è naturale fare dei conti mentalmente per prendere la banconota giusta e aspettarsi o meno il resto. Per poter eseguire calcoli, è indispensabile "rappresentare" le informazioni in modo che sia chiaro e comprensibile di che cosa si sta parlando.

I computer hanno un loro modo di rappresentare i dati che è diverso da quello usato quotidianamente dall'uomo: è un sistema basato sulle due sole cifre 0 e 1.

**Sistemi di numerazione**

Per **sistema di numerazione** si intende l'insieme dei simboli usati per rappresentare i numeri e le operazioni che possono essere eseguite su di essi.

Le varie civiltà che si sono susseguite nel corso dei secoli hanno avuto dei sistemi di numerazione diversi. A titolo di esempio, nella tabella sono riportati alcuni numeri del sistema romano e del sistema egizio.

| Sistema romano | Sistema egizio |
|---|---|
| I = 1 | 1 |
| V = 5 | 10 |
| X = 10 | 100 |
| L = 50 | 1.000 |
| C = 100 | 10.000 |
| D = 500 | 100.000 |
| M = 1.000 | 1 milione o infinito |

Con il sistema di numerazione usato abitualmente, quello decimale, è possibile combinare le dieci cifre da 0 a 9 per rappresentare tutti i numeri e realizzare ogni tipo di operazione. Nel sistema decimale la posizione di ciascuna cifra ha un determinato significato, per questo motivo è definito **sistema posizionale**. Per esempio nel numero 254, la cifra 4 indica le unità, il 5 le decine e il 2 le centinaia. In altre parole:

$254 = 2*100 + 5*10 + 4*1$

che è rappresentabile in questo modo tramite potenze di 10:

$254 = 2*10^2 + 5*10^1 + 4*10^0$

In questo caso specifico è stato usato il **metodo posizionale in base dieci**, in quanto le cifre vengono moltiplicate con opportune potenze del numero 10.

Con lo stesso metodo di ragionamento, si può definire un sistema di numerazione in base a qualsiasi numero: data la base, si moltiplica la cifra posizionale per la potenza della base.

**La rappresentazione in base binaria**

I calcolatori utilizzano un tipo di numerazione che sfrutta come base il numero 2: il **sistema binario**, in cui si usano solo due cifre: 0 e 1.

Per comprendere bene il perché sia stato necessario adottare tale sistema, va fatta una breve parentesi. Si supponga di avere un circuito elettrico collegato a una lampadi-

na, in modo che quando è chiuso, la lampadina si accende, altrimenti la lampadina rimane spenta.

Convenzionalmente si indica con il valore 1 la lampadina accesa e con 0 la lampadina spenta. Il sistema binario può rappresentare perfettamente questa situazione.

Il termine **bit** (da **bi**nary digi**t**, cioè cifra binaria) deriva proprio dal fatto che l'1 e lo 0 sono le cifre base del sistema binario. Combinando opportunamente i valori 0 e 1 è possibile rappresentare qualsiasi situazione definita da sequenze di circuiti collegati a lampadine.

Per esempio, i cinque circuiti attivi nei seguenti stati:

*chiuso, aperto, aperto, chiuso, aperto*

possono essere rappresentarli nel seguente modo:

10010

Il calcolatore, per quanto riesca a eseguire calcoli complessi e a elaborare dati strutturati, ha un principio di funzionamento molto semplice. Semplificando al massimo la sua struttura, può essere immaginato come un insieme di milioni e milioni di circuiti da cui può passare o meno un segnale elettrico.

La combinazione di accensione / spegnimento dei singoli circuiti dà luogo alle varie informazioni elaborate.

L'elemento di base della memoria è quindi sempre costituito da un'unità binaria, che può assumere solo due possibili stati: "mancanza di corrente" o "presenza di corrente", convenzionalmente rappresentati come 0 e 1.

Ogni circuito può assumere solo uno dei due stati possibili. Se si combinano due circuiti, aumentano le possibilità di informazioni fornite.

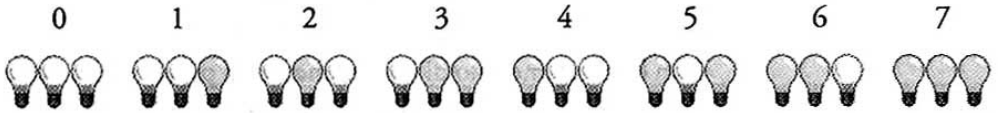
Si supponga di avere due lampadine, ognuna collegata a un circuito. Nella figura sono rappresentate le quattro possibilità di accensione/spegnimento:

*spento/spento* *acceso/spento* *spento/acceso* *acceso/acceso*

tradotti in bit:

0/0 1/0 0/1 1/1

Con la combinazione di tre circuiti si avranno otto combinazioni di bit e i numeri da 0 a 7 verrebbero rappresentati in questo modo:



0 1 2 3 4 5 6 7

000 001 010 011 100 101 110 111

Aumentando i circuiti, aumentano le possibili combinazioni di bit. Per esempio, se i circuiti sono tre i possibili stati sono:

*spento/spento/spento* *spento/spento/acceso*
*spento/acceso/spento* *spento/acceso/acceso*

Combinando ancora più bit, quindi, aumentano le possibilità di rappresentazione dei dati: con quattro bit si ottengono 16 combinazioni, con cinque 32, con sei 64, con sette 128, con otto 256 e così via.

I valori ottenuti sono sempre potenze di due e precisamente $16 = 2^4$, $32 = 2^5$, $64 = 2^6$, $128 = 2^7$ e $256 = 2^8$. Se ne può dedurre pertanto che da un numero $n$ di bit si ottengono $2^n$ combinazioni di stati e quindi $2^n$ possibili valori rappresentati.

**Rappresentazione dei caratteri**

Considerando combinazioni di otto bit si possono rappresentare quindi 256 possibili valori, sufficienti per rappresentare tutti i caratteri di cui si ha bisogno per scrivere testi e codici. Una combinazione di otto bit viene definita **Byte**, i cui valori danno luogo a 256 caratteri distinti (lettere, cifre, segni di interpunzione).

Questo sistema di codifica viene chiamato **ASCII** (acronimo di American Standard Code for Information Interchange, in italiano Standard americano per lo scambio di informazioni) e la corrispondenza byte / carattere è raccolta in una tabella standard definita *Tabella ASCII standard* (Figura 1.1).

**Figura 1.1:** Tabella ASCII Standard

| 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 |

La tabella è costituita da tre colonne: nella prima è riportato il byte, nella seconda il codice ASCII e nella terza il carattere o il simbolo associato.

In realtà lo standard ASCII ricopre solo i primi 128 byte (da 00000000 a 01111111). I rimanenti byte costituiscono quella che viene definita la *Tabella ASCII estesa* (Figura

**Figura 1.2:**
Tabella ASCII estesa

| Byte | Cod. | Char |
|---|---|---|
| 10000000 | 128 | Ç |
| 10000001 | 129 | ü |
| 10000010 | 130 | é |
| 10000011 | 131 | â |
| 10000100 | 132 | ã |
| 10000101 | 133 | à |
| 10000110 | 134 | å |
| 10000111 | 135 | ç |
| 10001000 | 136 | ê |
| 10001001 | 137 | ë |
| 10001010 | 138 | è |
| 10001011 | 139 | ï |
| 10001100 | 140 | î |
| 10001101 | 141 | ì |
| 10001110 | 142 | Ä |
| 10001111 | 143 | Å |
| 10010000 | 144 | É |
| 10010001 | 145 | æ |
| 10010010 | 146 | Æ |
| 10010011 | 147 | ô |
| 10010100 | 148 | ö |
| 10010101 | 149 | ò |
| 10010110 | 150 | û |
| 10010111 | 151 | ù |
| 10011000 | 152 | ÿ |
| 10011001 | 153 | Ö |
| 10011010 | 154 | Ü |
| 10011011 | 155 | ø |
| 10011100 | 156 | £ |
| 10011101 | 157 | Ø |
| 10011110 | 158 | × |
| 10011111 | 159 | ƒ |

| Byte | Cod. | Char |
|---|---|---|
| 10100000 | 160 | á |
| 10100001 | 161 | í |
| 10100010 | 162 | ó |
| 10100011 | 163 | ú |
| 10100100 | 164 | ñ |
| 10100101 | 165 | Ñ |
| 10100110 | 166 | ª |
| 10100111 | 167 | º |
| 10101000 | 168 | ¿ |
| 10101001 | 169 | ® |
| 10101010 | 170 | ¬ |
| 10101011 | 171 | ½ |
| 10101100 | 172 | ¼ |
| 10101101 | 173 | ¡ |
| 10101110 | 174 | « |
| 10101111 | 175 | » |
| 10110000 | 176 | ▒ |
| 10110001 | 177 | ▒ |
| 10110010 | 178 | ▓ |
| 10110011 | 179 | │ |
| 10110100 | 180 | ┤ |
| 10110101 | 181 | Á |
| 10110110 | 182 | Â |
| 10110111 | 183 | À |
| 10111000 | 184 | © |
| 10111001 | 185 | ╣ |
| 10111010 | 186 | ║ |
| 10111011 | 187 | ╗ |
| 10111100 | 188 | ╝ |
| 10111101 | 189 | ¢ |
| 10111110 | 190 | ¥ |
| 10111111 | 191 | ┐ |

| Byte | Cod. | Char |
|---|---|---|
| 11000000 | 192 | └ |
| 11000001 | 193 | ┴ |
| 11000010 | 194 | ┬ |
| 11000011 | 195 | ├ |
| 11000100 | 196 | ─ |
| 11000101 | 197 | ┼ |
| 11000110 | 198 | ã |
| 11000111 | 199 | Ã |
| 11001000 | 200 | ╚ |
| 11001001 | 201 | ╔ |
| 11001010 | 202 | ╩ |
| 11001011 | 203 | ╦ |
| 11001100 | 204 | ╠ |
| 11001101 | 205 | ═ |
| 11001110 | 206 | ╬ |
| 11001111 | 207 | ¤ |
| 11010000 | 208 | ð |
| 11010001 | 209 | Ð |
| 11010010 | 210 | Ê |
| 11010011 | 211 | Ë |
| 11010100 | 212 | È |
| 11010101 | 213 | ı |
| 11010110 | 214 | Í |
| 11010111 | 215 | Î |
| 11011000 | 216 | Ï |
| 11011001 | 217 | ┘ |
| 11011010 | 218 | ┌ |
| 11011011 | 219 | ▒ |
| 11011100 | 220 | ▒ |
| 11011101 | 221 | ▒ |
| 11011110 | 222 | Ì |
| 11011111 | 223 | ▒ |

| Byte | Cod. | Char |
|---|---|---|
| 11100000 | 224 | Ó |
| 11100001 | 225 | ß |
| 11100010 | 226 | Ô |
| 11100011 | 227 | Ò |
| 11100100 | 228 | õ |
| 11100101 | 229 | Õ |
| 11100110 | 230 | µ |
| 11100111 | 231 | þ |
| 11101000 | 232 | Þ |
| 11101001 | 233 | Ú |
| 11101010 | 234 | Û |
| 11101011 | 235 | Ù |
| 11101100 | 236 | ý |
| 11101101 | 237 | Ý |
| 11101110 | 238 | ¯ |
| 11101111 | 239 | ´ |
| 11110000 | 240 |  |
| 11110001 | 241 | ± |
| 11110010 | 242 | ‗ |
| 11110011 | 243 | ¾ |
| 11110100 | 244 | ¶ |
| 11110101 | 245 | § |
| 11110110 | 246 | ÷ |
| 11110111 | 247 | ¸ |
| 11111000 | 248 | ° |
| 11111001 | 249 | ¨ |
| 11111010 | 250 | · |
| 11111011 | 251 | ¹ |
| 11111100 | 252 | ³ |
| 11111101 | 253 | ² |
| 11111110 | 254 | ■ |
| 11111111 | 255 | |

1.2) che aggiunge tutti i simboli aritmetici, i simboli di valuta, le vocali accentate e altro. La sua struttura è identica a quella della tabella standard.

Il carattere o la parola o la frase che viene digitata viene quindi "trasformata" all'interno dell'elaboratore in una sequenza di byte. Per esempio la parola "vacanza" corrisponde alla seguente sequenza di byte:

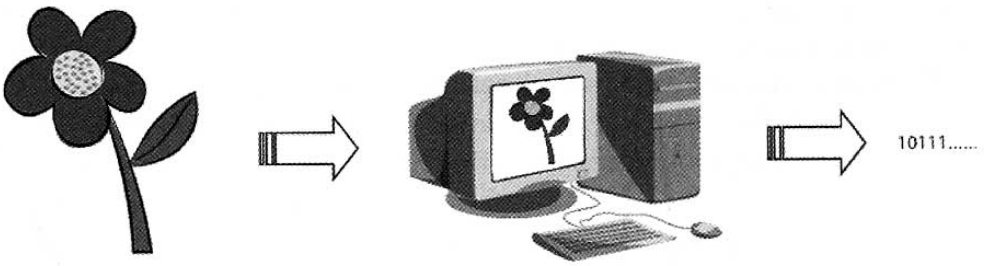
| v | a | c | a | n | z | a |
|---|---|---|---|---|---|---|
| 01110110 | 01100001 | 01100011 | 01100001 | 01101110 | 01111010 | 01100001 |

**Rappresentazione di immagini**

Anche le immagini vengono rappresentate con sequenze di bit, ma la situazione cambia radicalmente.

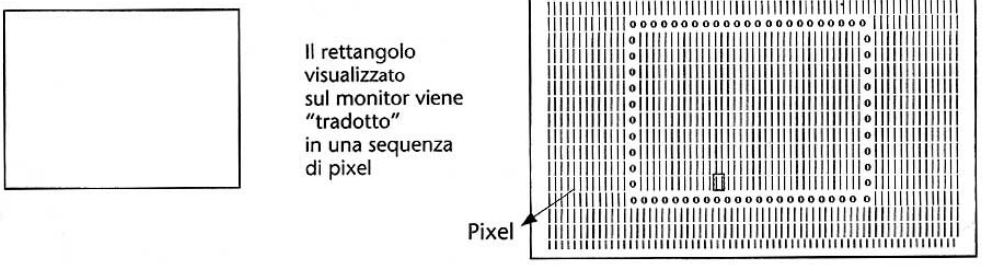
Prima di procedere con l'elaborazione di un'immagine, è necessario che venga trasformata in un'informazione "discreta", cioè costituita da insiemi di parti distinte, ciascuna delle quali può essere elaborata come sequenza di bit.

L'immagine deve essere scomposta in una sequenza di punti, ognuno rappresentato da uno o più bit.




Tali punti sono chiamati **pixel**, (contrazione di **pic**ture **el**ement).

Per esempio: si supponga che sul monitor del computer sia stato tracciato un rettangolo. Per poterlo elaborare, il computer deve scomporlo in una griglia di elementi, come quella riportata in figura.




Ogni pixel quindi viene convertito in uno o più bit a seconda delle informazioni sul colore che contiene. Se l'immagine deve essere rappresentata in tonalità di grigio può essere sufficiente un byte per ciascun pixel (8 bit), che diventano tre se dobbiamo rappresentarla a colori. Il numero di colori possibili, pertanto, dipende dalla quantità di bit utilizzata: un'immagine con 1 bit per pixel avrà al massimo due combinazioni possibili (0 e 1) e quindi potrà rappresentare solo due colori; nelle immagini a 4 bit per pixel, si potranno rappresentare al massimo 16 colori; in immagini a 8 bit per pixel se ne avranno 256 e così via. Nelle immagini a colori, quindi, ogni pixel ha una sua luminosità e colore, tipicamente rappresentati da una tripletta di intensità di rosso, verde e blu, detta RGB (Red, Green, Blue). In breve:

| Tipo di immagine | Bianco/nero | Scala di grigi | Colore RGB |
|---|---|---|---|
| Bit per pixel | 1 | 8/16 | 24/48 |

In sostanza, se aumenta il numero di bit usati per rappresentare un pixel aumentano anche le informazioni sul colore, rendendo l'immagine più fedele all'originale. Come conseguenza, però, aumenterà anche lo spazio che essa occuperà in memoria.

**Rappresentazione di suoni**

Un suono può essere rappresentato graficamente tramite una linea continua che si alza e si abbassa nel tempo, secondo i toni del suono stesso (Figura 1.3).

Così come è avvenuto per le immagini, è necessario suddividere il tempo in intervalli regolari e individuare il valore del suono in quel preciso istante. Tale operazione viene definita **campionamento** e viene creato come risultato un insieme di punti, detti **campioni** (Figura 1.4).

**Figura 1.3:** Rappresentazione di un suono tramite un grafico

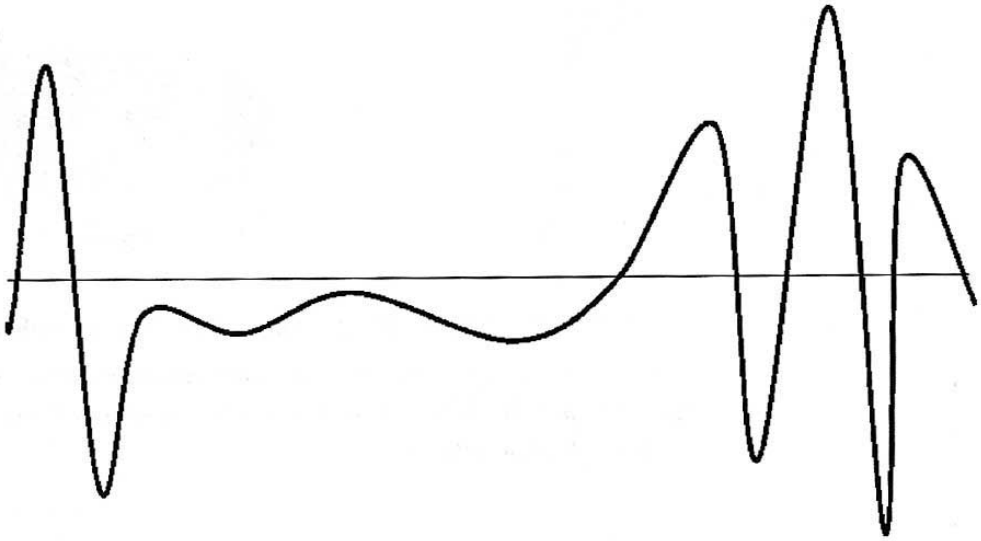
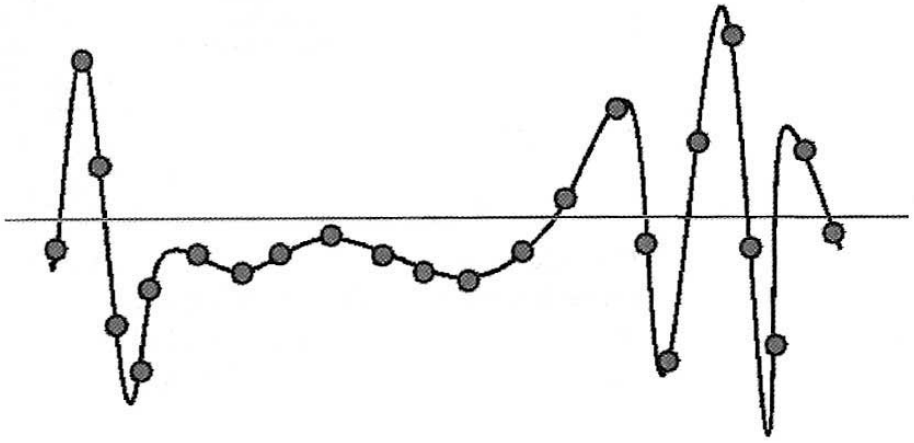


**Figura 1.4:** Rappresentazione di un suono tramite un campionamento



Ciascun campione può essere rappresentato tramite un'opportuna sequenza di bit.

Come per l'immagine, quanto più alto è il numero di campioni, maggiore è la fedeltà al suono originale, e maggiore è la quantità di spazio occupato in memoria.

Per diminuire lo spazio di memoria occupata vengono applicate opportune tecniche di **compressione**, una delle quali è quella usata per i tanto ascoltati brani in formato MP3. In tal caso viene sfruttato il fatto che l'orecchio umano non riesce a percepire suoni a basso volume sovrapposti a suoni ad alto volume, che quindi vengono eliminati ottimizzando l'occupazione di memoria.

# Linguaggi

**Il linguaggio macchina**

In generale, un linguaggio è un sistema di simboli combinati secondo determinate regole con cui più soggetti che "parlano la stessa lingua" possono comunicare. Si pensi al linguaggio con cui quotidianamente le persone comunicano tra loro. La comunicazione è possibile perché tutti sono in grado esprimere concetti tramite parole. La reciproca comprensione è aiutata anche dalle espressioni del viso e dalla gestualità.

Al contrario, i computer basano tutta la loro comunicazione e rappresentazione dei dati tramite il sistema binario. Tutti i programmi eseguibili dagli elaboratori sono scritti con un linguaggio, chiamato **linguaggio macchina**, basato sui simboli 1 e 0. Questi simboli sono organizzati in "parole" che a loro volta costituiscono "frasi", chiamate **istruzioni**: ognuna indica un'azione elementare da eseguire, come la lettura di un dato o l'esecuzione della somma di due numeri.

Due "frasi" possono essere collegate tra loro tramite operatori, detti **connettivi logici**, per dare luogo a una terza "frase". I più importanti sono **AND** e **OR**. Due frasi connesse tramite AND restituiscono una frase vera solo se sono entrambe vere, altrimenti danno luogo a una frase falsa.

Se si connettono due frasi tramite l'OR, si ottiene una frase vera se almeno una delle due è vera.

Per esempio, si prendano le due frasi "Pisa è in Toscana" e "la Toscana confina con la Lombardia".

Se vengono connesse tramite l'operatore AND, si ottiene una frase falsa (*Pisa è in Toscana AND la Toscana confina con la Lombardia*). Se vengono connesse con l'operatore OR, si ottiene un frase vera (*Pisa è in Toscana OR la Toscana confina con la Lombardia*).

Un altro connettivo logico è il NOT, che permette di ottenere la negazione della frase. La negazione della seconda frase (NOT *la Toscana confina con la Lombardia*) è vera.

**I linguaggi simbolici**

Visto che il linguaggio naturale usato dall'uomo è molto più complesso del linguaggio macchina, è necessario avere a disposizione uno strumento che permetta al computer di "comprendere" i comandi impartiti. A questo scopo sono nati i **linguaggi di programmazione** (detti anche linguaggi simbolici) che permettono la comunicazione tra uomo e computer. Le istruzioni scritte con tali linguaggi vengono tradotte in codice binario (e viceversa) tramite un apposito programma chiamato **compilatore**. Tra i linguaggi di programmazione più diffusi possono essere citati **C**, **C++**, **Visual Basic** e **Java**.

**Descrizione di algoritmi mediante uno pseudolinguaggio**

Si è visto come gli algoritmi possano essere rappresentati graficamente tramite diagrammi di flusso. È anche possibile rappresentarli in modo testuale tramite pseudolinguaggi, cioè con dei linguaggi di programmazione "fittizi", non direttamente interpretabili dal compilatore. Scrivere un programma tramite pseudocodice permette di concentrarsi più sul funzionamento dell'algoritmo e sulle istruzioni da seguire che sul formalismo del linguaggio di programmazione.

I passaggi da compiere sono:

- trasformare un algoritmo in uno pseudocodice;
- scegliere il linguaggio di programmazione più adatto all'applicazione;
- trasformare i vari comandi dello pseudocodice in comandi scritti con il linguaggio di programmazione scelto.

Per esempio, il diagramma di flusso che riguarda l'acquisto di tre maglie, due camicie e una gonna, può essere trasformato nel seguente pseudocodice.

```
1.  BEGIN
2.  READ x
```

```
3.  READ y
4.  x*3=costomaglie
5.  y*2=costocamicie
6.  costomaglie+costocamicie=costototale
7.  70,00-costototale=soldirimanenti
8.  IF soldirimanenti>=12,00 THEN soldirimanenti-
    costogonna=resto ELSE soldirimanenti=resto
9.  END
```

Il significato in dettaglio di queste istruzioni è il seguente.

Con `BEGIN` ("inizio") e `END` ("fine") si indica l'inizio e la fine del programma (righe 1 e 9).

Con il comando `READ` ("leggi") si legge il prezzo delle maglie (x) e delle camicie (y) (righe 2 e 3).

Alla riga 4, 5 e 6 vengono calcolati rispettivamente i costo per l'acquisto delle maglie e per le camicie e vengono sommati.

La riga 7 serve per calcolare quanti soldi rimangono: dai 70,00 € deve essere sottratta la spesa per l'acquisto di maglie e camicie.

Per essere sicuri di poter comprare anche la gonna, va verificato quanto denaro resta a disposizione. Il comando `IF` ("se") della riga 8 serve proprio a questo: se rimangono almeno 12,00 €, può essere acquistata anche la gonna e si riceve il resto (ramo `THEN`), altrimenti è necessario rinunciare e ricevere solo il resto (ramo `ELSE`). Questa riga corrisponde al simbolo rombo del diagramma di flusso: è un'istruzione che implica una scelta tra due possibili percorsi a seconda del valore restituito dalla valutazione di una certa condizione (in questo caso il denaro rimanente).