CADEL

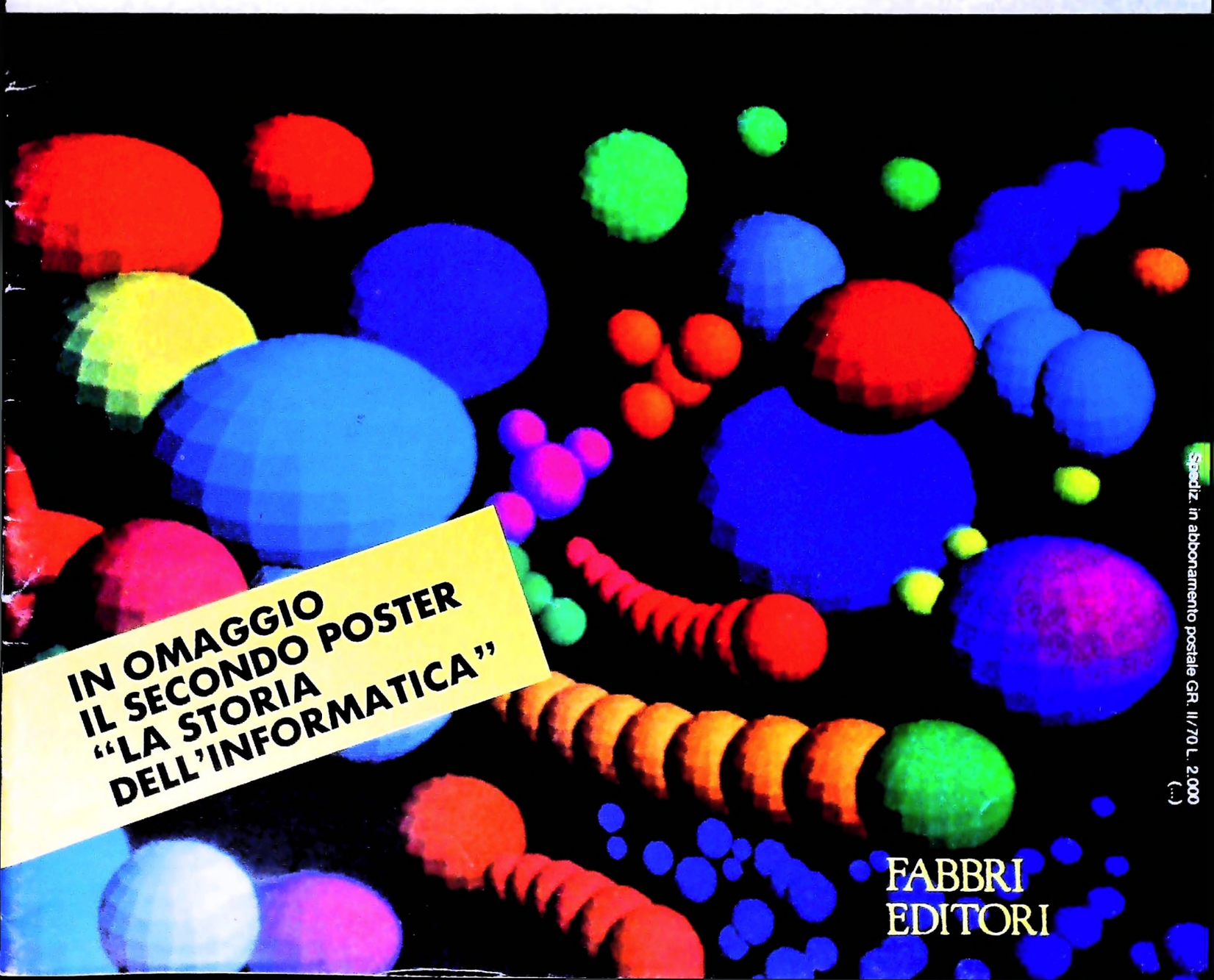
# 7 CORSO PRATICO COL COMPUTER

421594

è una iniziativa
**FABBRI EDITORI**
in collaborazione con
**BANCO DI ROMA**
e **OLIVETTI**

F4 F5 F6 F7 F8 BATTERY LOW.

diretto da GIANNI DEGLI ANTONI

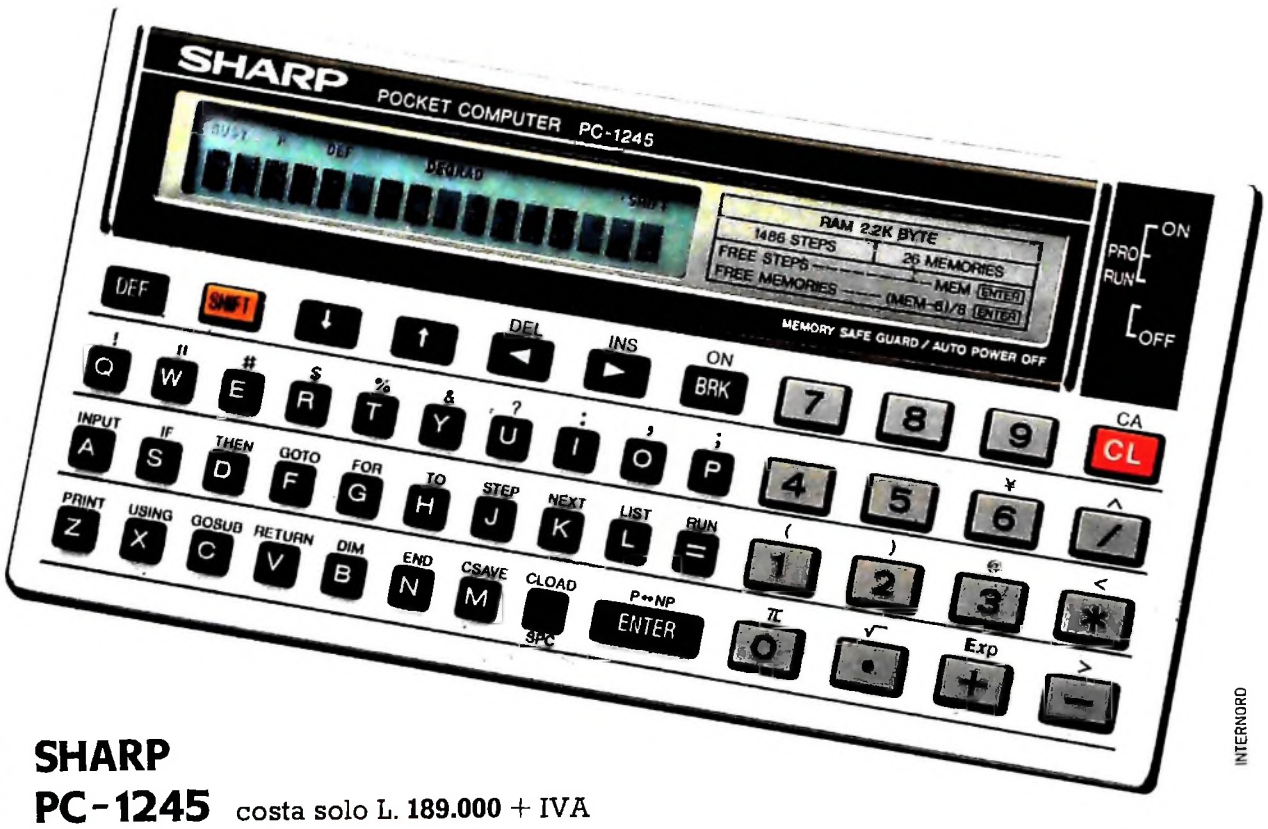


**IN OMAGGIO IL SECONDO POSTER "LA STORIA DELL'INFORMATICA"**

FABBRI EDITORI

Spediz. in abbonamento postale GR. II/70 L. 2.000 (...)

# I tasti predefiniti con i comandi in Basic rendono ancora più facile la programmazione dando al neofita un piccolo grande aiuto.




INTERNORD

## SHARP PC-1245 costa solo L. 189.000 + IVA

**Facile inserimento dei più usati comandi in BASIC.**
Il sistema d'impostazione istantanea dei comandi in BASIC facilita notevolmente sia l'operatività del Computer che la programmazione dello stesso.
I 18 tasti alfabetici sono preprogrammati con i più comuni comandi BASIC; ad esempio al tasto A è abbinato il comando INPUT, al tasto F il comando GOTO ed al tasto Z il comando PRINT e così via. Questo, durante la programmazione, evita di dover comtinuamente scrivere i vari comandi per intero, garantendovi meno errori, una maggior velocità nella programmazione e facilitandovi l'apprendimento dei termini in BASIC.

**18 tasti definibili dall'utilizzatore per etichettare i programmi**
Questa è una interessante possibilità che vi permette di accedere immediatamente ai programmi più usati. Potete etichettare fino a 18 programmi assegnando loro un tasto. Per richiamare un programma basta premere il tasto assegnato.

**Potenza portatile a vostra disposizione.**
Potrete avere la potenza del Computer ovunque vi serve.

Il PC-1245 ha una capacità di 24 KBytes di ROM per governare l'intero sistema e 2,2 KBytes di RAM per programmazione.

**C-MOS CPU a 8-bit**
Il PC-1245 usa la stessa CPU dei Personal da tavolo.
Questo, oltre ad una alta velocità d'elaborazione, vi garantisce una grande efficienza.

**Tastiera tipo macchina da scrivere**
Grazie alla disposizione dei tasti come sulle macchine da scrivere vi sarà facile impostare velocemente i vostri programmi.

**Memoria protetta**
Uno speciale sistema d'alimentazione protegge la memoria del PC-1245 anche a macchina spenta. Questo vi consente d'interrompere, in qualsiasi momento, un programma od un calcolo. Potete ricominciare quando volete dall'ultimo inserimento - anche dopo giorni - senza correre il rischio d'aver perso un dato o una istruzione.

**Funzione PASS**
Potete assegnare un codice segreto al programma in memoria ottenendo così una completa protezione dello stesso. Non sarà possibile listarlo, modificarlo o vederlo. Si potrà solo elaborare.

**Visore a 16 caratteri con matrice a punti 5x7**
Il visore è in grado di visualizzare contemporaneamente fino a 16 caratteri. Ogni carattere appare chiaramente leggibile grazie alla matrice a punti di 5 per 7. Potete anche regolare la luminosità del visore per avere la miglior lettura.

**Selezione tra RUN e programma**
Un interruttore consente l'immediata selezione tra il modo Run e PROGRAMMA.

**Un optional importante**
Per dare al vostro programma una chance in più potete integrare il PC-1245 con la CE-125, stampante e microregistratore opzionale, rendendo il sistema ancora più completo. La possibilità di stampare e registrare su nastro i vostri programmi e dati vi sarà utile per conservare sia i risultati della elaborazione che i programmi realizzati.
La CE-125 contiene armoniosamente il PC-1245 mantenendo le dimensioni di un libro.

concessionaria per l'Italia
**MELCHIONI**

**TUTTA LA POTENZA DI UN COMPUTER NEL PALMO DELLA TUA MANO**

Per ulteriori informazioni scrivete a
MELCHIONI - Divisione Pocket Computer - 20135 MILANO - Via P. Colletta 37

SHARP


Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'Informazione, Direttore dell'Istituto di Cibernetica dell'Università degli Studi di Milano

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

MARIO ITALIANI
Ordinario di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

MARCO MAIOCCHI
Professore Incaricato di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

Curatori di rubriche
TULLIO CHERSI, ADRIANO DE LUCA (Professore di Architettura dei Calcolatori all'Università Autonoma Metropolitana di Città del Messico), GOFFREDO HAUS, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI, ENNIO PROVERA

Testi
GOFFREDO HAUS, Eidos (TIZIANO BRUGNETTI), ENNIO PROVERA, VIRGINIA SALA, Etnoteam (ADRIANA BICEGO)

Tavole
Logical Studio Communication
Il Corso di Programmazione e BASIC è stato realizzato da Etnoteam S.p.A., Milano
Computergrafica è stato realizzato da Eidos, S.c.r.l., Milano
Usare il Computer è stato realizzato in collaborazione con PARSEC S.N.C. - Milano

Direttore Editoriale
ORSOLA FENGHI

Coordinatore settore scientifico
UGO SCAIONI

Redazione
MARINA GIORGETTI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI


È in edicola il secondo numero di LIBRERIA DI SOFTWARE dedicato alla STORIA MEDICA FAMILIARE: il listato ed il programma già pronto per Olivetti M10, COMMODORE 64 e ZX SINCLAIR SPECTRUM

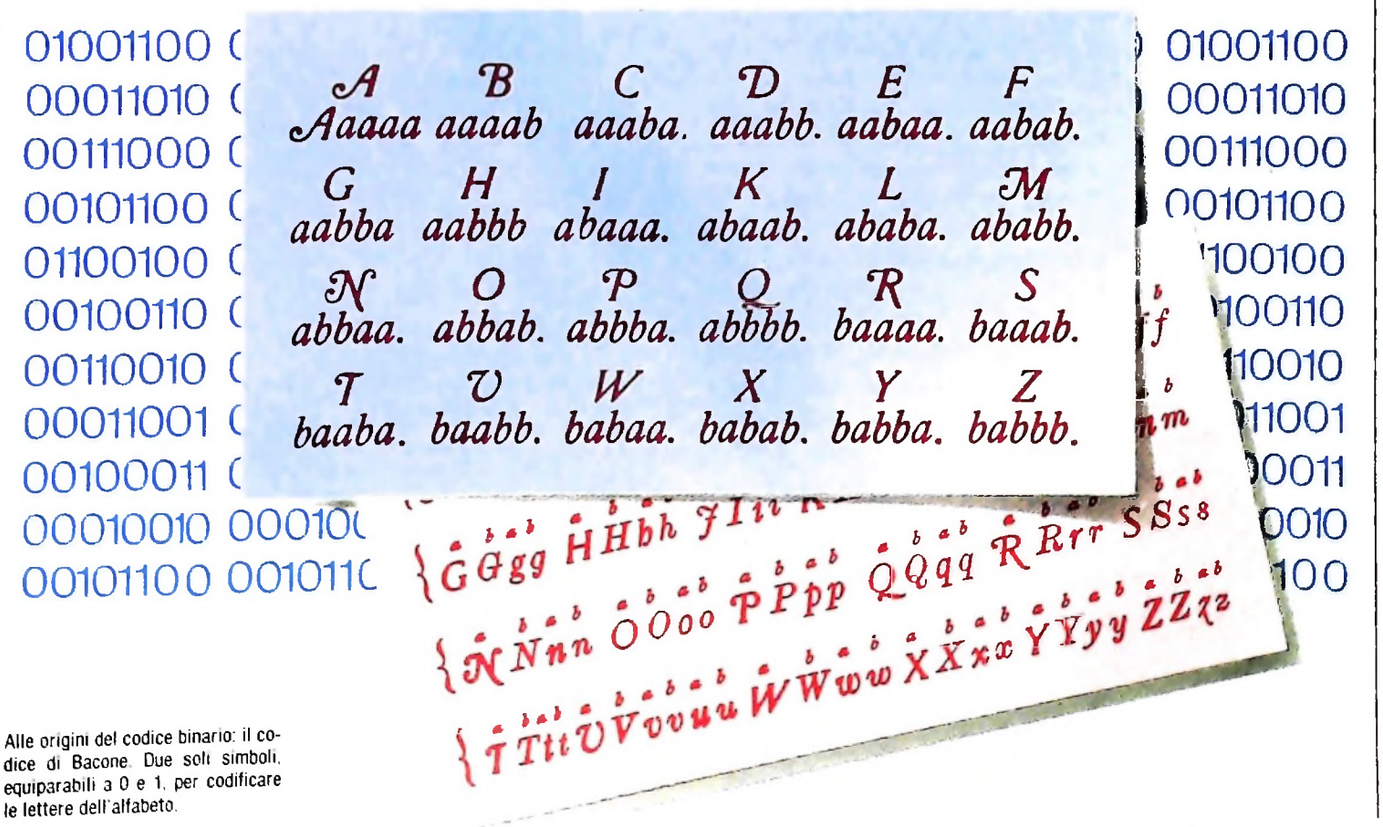
# Sistemi di numerazione

## Binario, ottale, decimale, esadecimale: modi diversi ma equivalenti di rappresentare i numeri.

Siete in una sala di controllo e davanti a voi sta una lampadina rossa: finché è spenta tutto va bene, se si accende significa che è intervenuta una situazione di pericolo. Per rappresentare questo sistema bastano due cifre: uno 0 e un 1, 0 per la lampadina spenta, 1 per la lampadina accesa.
Ora la sala di controllo è cambiata, e le lampadine sono due: una rossa indica, se accesa, un guasto all'impianto controllato, una blu indica, se accesa, che qualcuno è entrato in una zona proibita. Con lo stesso criterio di prima, bastano quattro coppie di cifre per identificare qualunque situazione: 00 per dire che le due lampadine sono spente; 01 per rossa spenta, blu accesa; 10 per rossa accesa, blu spenta; 11 per tutte e due le lampadine accese.
È ora facile immaginare per analogia come si possa rappresentare qualunque situazione che coinvolga un insieme di lampadine, di interruttori, o più genericamente di oggetti che possono assumere due soli stati (acceso o spento, aperto o chiuso): e un calcolatore non è altro, in fin dei conti, che un enorme insieme di componenti elementari caratterizzati da due soli stati: passa corrente o non passa corrente. Un "codice binario" come quello usato per rappresentare le nostre lampadine è lo strumento che si offre immediatamente alla nostra attenzione per la sua descrizione. Viceversa, quando dobbiamo comunicare qualcosa al calcolatore, in ultima istanza quello che deve arrivare alla macchina è una sequenza di 0 e 1 ("cifre binarie" o "bit"), ed è inevitabile dover avere un modo per esprimere con queste due sole cifre ogni simbolo, e in particolare i numeri.

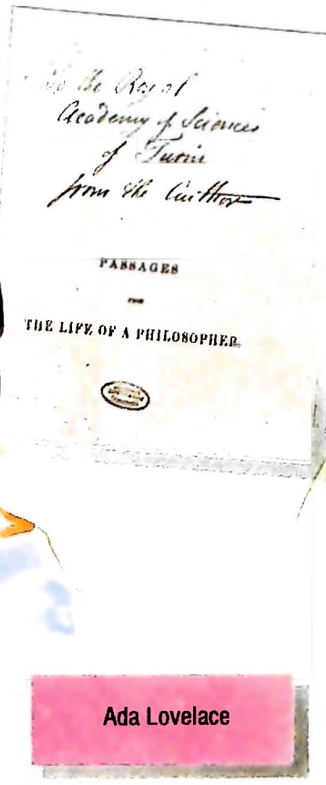
### Il sistema di numerazione binario

Intorno al Seicento, il codice binario attirò l'interesse di molti come codice cifrato: ma il suo valore supera di molto quello che gli può essere accreditato come strumento di segretezza, di spionaggio o d'intrigo.




Alle origini del codice binario: il codice di Bacone. Due soli simboli, equiparabili a 0 e 1, per codificare le lettere dell'alfabeto.

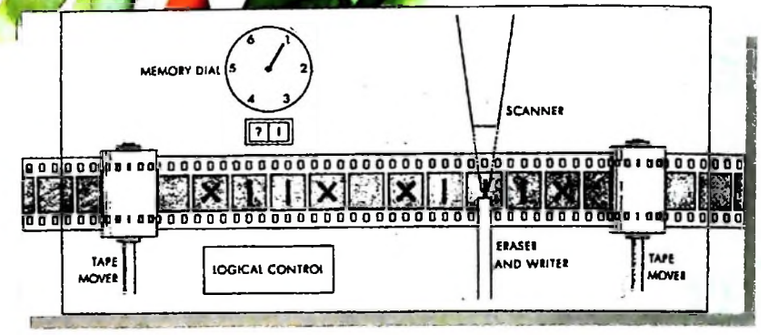
Lady Ada Lovelace, figlia di Lord Byron: la prima programmatrice, per la collaborazione prestata a Charles Babbage.




**Ada Lovelace**

Alan Turing, negli anni Trenta del nostro secolo, formulò alcuni concetti centrali per l'informatica teorica: la "macchina di Turing" è un modello ideale di calcolatore.

**Alan Turing**




Con due cifre — 0 e 1 — si possono rappresentare tutti i numeri: il sistema di numerazione binario ha una sua storia al di là del mondo dei calcolatori. I principi su cui si basa non sono diversi da quelli del sistema decimale: rivediamoli quindi brevemente nel sistema che ci è più familiare.

Il sistema di numerazione che usiamo tutti i giorni si dice decimale perché è basato sul numero dieci e fa uso di dieci cifre — da 0 a 9 — ed è un sistema posizionale: il valore di una cifra dipende dalla sua posizione. Quando scriviamo 147 vogliamo dire, in effetti, "1 centinaio, 4 decine e 7 unità", e 147 è diverso da 174 (anche se le cifre in gioco sono le stesse), ed è diverso anche da 1470 (benché lo zero indichi... nulla).

Chiariamoci meglio le idee su che cosa significhi "essere basato sul numero dieci". Un numero come 147 è una forma abbreviata per $1 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$. Ogni posizione in un numero di più cifre indica una potenza di dieci, crescente da destra verso sinistra (dieci elevato alla potenza 0 è, per convenzione, uguale a 1). Siamo così abituati a usare

| dec. | binario | ottale | esadec. |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |
| 16 | 10000 | 20 | 11 |
| 17 | 10001 | 21 | 12 |
| 18 | 10010 | 22 | 13 |
| 19 | 10011 | 23 | 14 |
| 20 | 10100 | 24 | 15 |

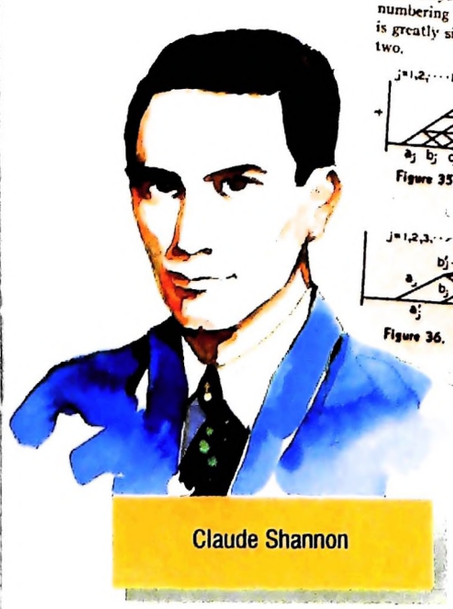
A lato, tabella dell'equivalente delle rappresentazioni dei primi numeri interi nei sistemi di numerazione decimale, binario, ottale ed esadecimale.

**Tabella dell'addizione ottale**

| + | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 |
| 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 |
| 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 |
| 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 |
| 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

**Tabella della moltiplicazione ottale**

| × | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 2 | 4 | 6 | 10 | 12 | 14 | 16 |
| 3 | 3 | 6 | 11 | 14 | 17 | 22 | 25 |
| 4 | 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 7 | 16 | 25 | 34 | 43 | 52 | 61 |



Claude Shannon

A Claude Shannon si devono i concetti fondamentali della teoria dell'informazione.

ELECTRIC ADDER TO THE BASE TWO

A circuit is to be designed that will automatically add two numbers, using only relays and switches. Although any numbering base could be used the circuit is greatly simplified by using the scale of two.

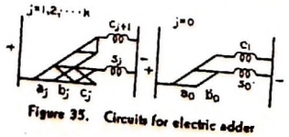


Figure 35. Circuits for electric adder

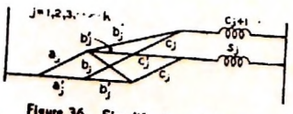
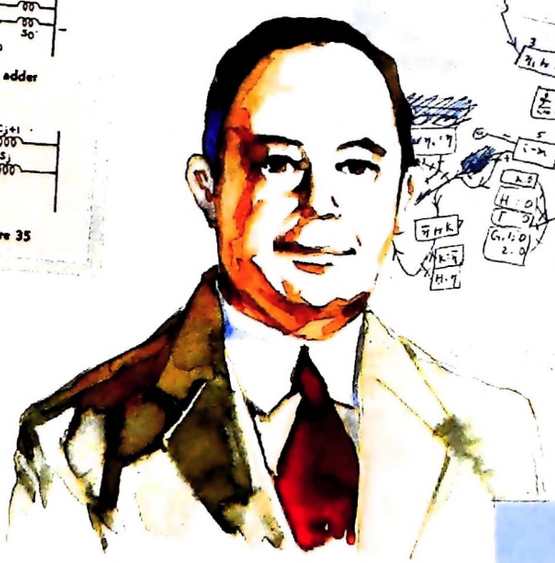
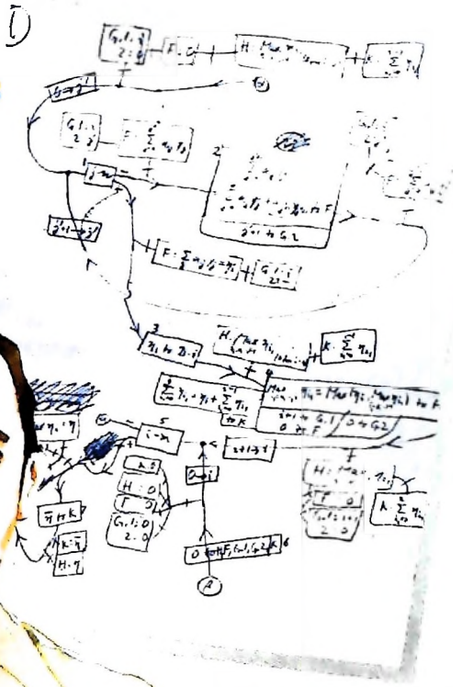


Figure 36. Simplification of figure 35





A John von Neumann si deve l'architettura moderna del calcolatore, con unità di ingresso e uscita, di elaborazione e di memoria.

John von Neumann

**Tabella dell'addizione esadecimale**

| + | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 |
| 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 |
| 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 |
| 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A |
| C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B |
| D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C |
| E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D |
| F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

**Tabella della moltiplicazione esadecimale**

| × | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 2 | 2 | 4 | 6 | 8 | A | C | E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E |
| 3 | 3 | 6 | 9 | C | F | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D |
| 4 | 4 | 8 | C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C |
| 5 | 5 | A | F | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B |
| 6 | 6 | C | 12 | 18 | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A |
| 7 | 7 | E | 15 | 1C | 23 | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 |
| 8 | 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| 9 | 9 | 12 | 1B | 24 | 2D | 36 | 3F | 48 | 51 | 5A | 63 | 6C | 75 | 7E | 87 |
| A | A | 14 | 1E | 28 | 32 | 3C | 46 | 50 | 5A | 64 | 6E | 78 | 82 | 8C | 96 |
| B | B | 16 | 21 | 2C | 37 | 42 | 4D | 58 | 63 | 6E | 79 | 84 | 8F | 9A | A5 |
| C | C | 18 | 24 | 30 | 3C | 48 | 54 | 60 | 6C | 78 | 84 | 90 | 9C | A8 | B4 |
| D | D | 1A | 27 | 34 | 41 | 4E | 5B | 68 | 75 | 82 | 8F | 9C | A9 | B6 | C3 |
| E | E | 1C | 2A | 38 | 46 | 54 | 62 | 70 | 7E | 8C | 9A | A8 | B6 | C4 | D2 |
| F | F | 1E | 2D | 3C | 4B | 5A | 69 | 73 | 87 | 96 | A5 | B4 | C3 | D2 | E1 |

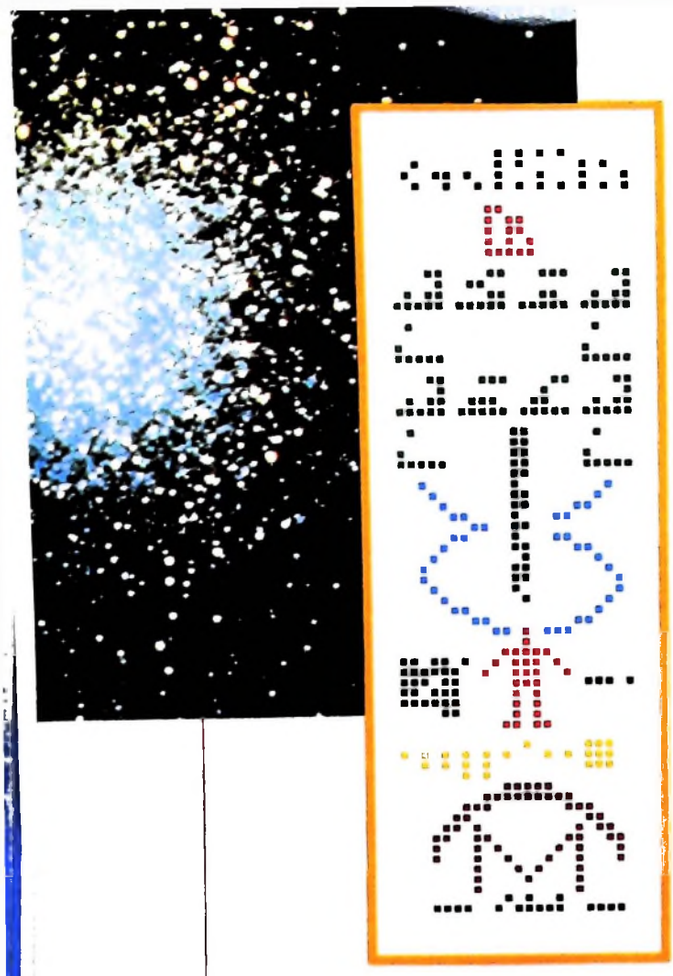
Le tabelle che vediamo raffigurate in queste due pagine danno le tavole di addizione e moltiplicazione per i sistemi ottale ed esadecimale: si usano come la tavola pitagorica del sistema decimale.

Ogni numero nelle tabelle rappresenta la somma (oppure, rispettivamente, il prodotto) dei due numeri che contrassegnano la riga e la colonna al cui incrocio si trova.

quotidianamente questo sistema che non abbiamo più bisogno di riflettere sulla sua caratteristica di fondo: ma ogni insegnante elementare sa bene quanto costa insegnare a un bambino a usare la notazione posizionale.

Il principio può essere esteso a qualunque "base" diversa dal dieci: mantenendo inalterato il valore posizionale delle cifre, si possono assumere le potenze di un numero qualunque come valori delle successive posizioni. Storicamente, accanto alla base dieci, hanno avuto uso pratico la base 60 (la usiamo ancora per i minuti e i secondi), la base 8 e la base 16 (usata dai babilonesi, per esempio); nella tipografia le lunghezze delle righe si misurano ancora oggi con un sistema tradizionale basato sul 12: 12 "punti" costituiscono una "riga" tipografica. Sul piano teorico, nulla vieta di costruire un sistema di numerazione con qualunque base: la scelta è motivata piuttosto da esigenze pratiche.

Per il sistema binario (che ha forti motivazioni, come abbiamo visto, nel campo dei calcolatori), la base è il due, e le uniche cifre sono 0 e 1. Seguendo lo stesso principio del sistema decimale, un numero come 100110 significa $1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$. Cioè, facendo un po' di calcoli (nel sistema decimale), 32 + 4 + 2, e quindi 38. Tutta la difficoltà nel maneggiare il sistema binario sta nel ricor-




Il "messaggio di Arecibo", inviato dal radiotelescopio di Arecibo nell'isola di Puerto Rico il 16 novembre 1974, diretto a eventuali abitanti intelligenti della Galassia, con affiancata la corrispondente versione in codice binario.

dare i valori delle potenze crescenti di 2: ma basta un po' d'esercizio.

Disponendo di due sole cifre, il sistema binario ci offre poche combinazioni fondamentali da ricordare, per eseguire le comuni operazioni come somma, sottrazione, prodotto e divisione: le "tabelline" del sistema binario sono molto semplici. Prendiamo la somma. I casi fondamentali sono solo quattro: 0 + 0 ci dà ancora 0; 0 + 1 e 1 + 0 ci danno come risultato 1; 1 + 1 ci dà come risultato 10 (ovvero "zero con il riporto di 1"). Qualunque somma con numeri a più cifre è solo la ripetuta applicazione di questi quattro casi, con lo stesso meccanismo o "algoritmo" ben noto nel caso decimale. Per la sottrazione la situazione non è diversa: i quattro casi fondamentali sono 0 - 0 = 0, 1 - 1 = 0, 1 - 0 = 1 e 0 - 1 = 1 con un "prestito" dalla cifra più a sinistra.

La moltiplicazione si riduce alla somma di più copie del moltiplicando, opportunamente incolonnate (le moltiplicazioni per 1 danno il moltiplicando, quelle per 0 danno ancora 0 e determinano solo lo spostamento di una posizione verso sinistra). Anche la divisione, infine, non offre sorprese: non c'è bisogno mai di calcoli complessi, perché le cifre del quoziente possono essere solo 0 e 1!

## I sistemi ottale ed esadecimale

Usare il sistema binario non presenta quindi grosse difficoltà di principio: tuttavia il numero delle cifre binarie necessarie per rappresentare un numero cresce rapidamente: per un numero che nel sistema decimale è rappresentabile con 4 cifre nel sistema binario occorrono da 9 a 13 cifre. Presto la semplificazione nelle quattro operazioni fondamentali è più che compensata dalla lunghezza delle rappresentazioni. Per la macchina questo non è un problema, ma è sicuramente una difficoltà per noi. Per questo spesso vengono usati due altri sistemi di numerazione, diversi da quello decimale, ma che intrattengono uno stretto rapporto con il sistema binario: l'ottale (a base 8) e l'esadecimale (a base 16). Poiché sia 8 che 16 sono potenze di 2, questi sistemi si riconducono facilmente al binario; sfruttando rispettivamente 8 e 16 cifre fondamentali, ci offrono d'altra parte rappresentazioni compatte come quelle del sistema decimale: hanno, cioè, alcuni pregi di tutti e due i sistemi. Per evitare confusioni tra i vari sistemi, adottiamo una comune convenzione: ove possano sorgere ambiguità, aggiungeremo a un numero un indice, che ne indica la base. Per esempio $10_{10}$ indica il numero dieci nel sistema decimale; $10_2$ è il due nel sistema binario.

Il sistema ottale, a base otto, usa otto cifre: si impiegano normalmente le cifre da 0 a 7. In questo sistema 0, 1, 2, 3, 4, 5, 6 e 7 hanno lo stesso significato che hanno nel sistema decimale, ma $8_{10}$ verrà poi rappresentato da $10_8$ (notate che il simbolo 10 indica, in tutti i sistemi di numerazione, la base del sistema stesso); 11 sta per 9, e così via.

Nel sistema esadecimale, a base 16, i simboli debbono essere 16: comunemente, accanto alle cifre da 0 a 9 si impiegano le lettere A, B, C, D, E, F (che corrispondono rispettivamente ai numeri 10, 11, 12, 13, 14, 15 del sistema decimale). La notazione risulta molto compatta ma anche un po' meno perspicua, per noi: A3F per esempio corrisponde a 10 x 16 + 3 x 16 + 15 x 16, cioè 2560 + 48 + 15 = 2623.

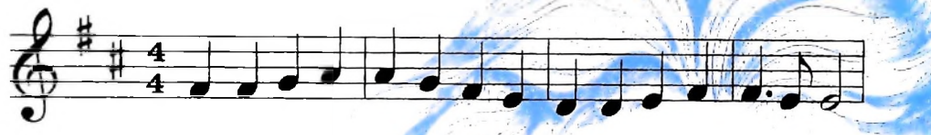
Le quattro operazioni fondamentali non presentano novità in nessuno di questi sistemi: con il numero più elevato di simboli, è inevitabile tuttavia far ricorso agli equivalenti della tavola pitagorica.

# LA MELODIA

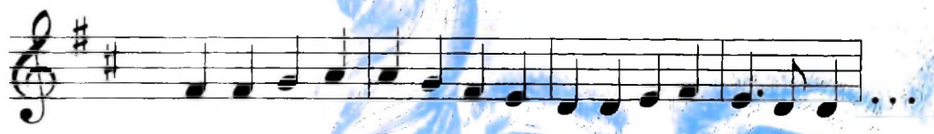
## La generazione automatica di melodie è tra le applicazioni musicali dei computer quella che conta più seguaci.

Il concetto di *melodia* musicale non è certamente il concetto riduttivo di sequenza di note; la melodia è una sequenza di note solo dal punto di vista della struttura che permette di descriverla. L'attributo che forse meglio caratterizza una melodia è la sua *tensione* interna, realizzata mediante un'opportuna sequenza di intervalli, ascendenti e discendenti, più o meno ampi. In precedenza abbiamo parlato delle scale musicali; certo le scale sono sequenze di altezze, ma non possiamo considerarle melodie per la banalità della loro sequenza di intervalli (tutti intervalli di ampiezza minima, tutti verso l'acuto se la scala è ascendente, tutti verso il grave se la scala è discendente). I tentativi noti di studiare e classificare le melodie musicali non hanno portato a risultati che possano essere ritenuti generali; tuttavia possiamo scegliere una impostazione semplice per distinguere tra i diversi tipi di melodie; per esempio, possiamo considerare i tre tipi di melodie seguenti:

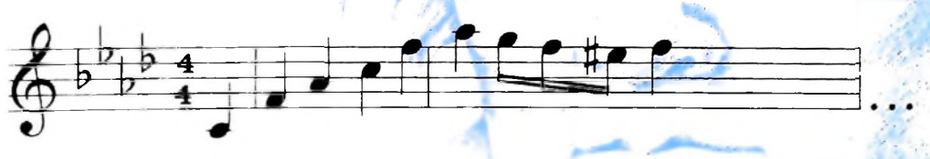
a) melodie per *gradi congiunti*, dove gli intervalli della melodia sono ottenuti mediante la composizione di sottosequenze della scala di riferimento (diatonica, temperata, pentatonica



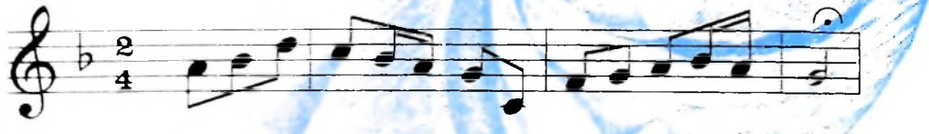
Esempio di melodia del tipo a, o per gradi congiunti, (tema dall'ultimo movimento della Nona Sinfonia di Ludwig van Beethoven.



Esempio di melodia del tipo b (dalla Sonata per pianoforte, op. 2, n. 1 di L. van Beethoven).



Esempio di melodia del tipo c (dalla Sesta Sinfonia "Pastorale" di L. van Beethoven).



o altre), ovvero mediante sequenze di intervalli minimi, come intervalli di seconda ascendenti o discendenti;
b) melodie per *gradi di terza, quinta* o intervalli ancora più ampi, in cui le sequenze di intervalli sono costituite da salti ampi nella scala a cui si fa riferimento;
c) melodie che sono una combinazione di melodie del tipo a) e del tipo b).

## Generazione automatica di melodie

Tra le applicazioni musicali degli elaboratori che hanno fino ad ora avuto più seguaci, figura sicuramente la generazione automatica di melodie. Gli studi di Rader, di Zaripov, di Grossi, di Baroni e Jacobini e di tanti altri ancora mirano appunto alla definizione di modelli della struttura musicale delle melodie, in modo da poter formalizzare la descrizione delle melodie e renderla operativa mediante programmi che la eseguono; l'obiettivo è di riuscire a descrivere formalmente la *grammatica della melodia*; le diverse impostazioni si distinguono essenzialmente per la profondità con cui sono considerati i diversi livelli di aggregazione del materiale melodico e per il riferimento o meno a una specifica prassi musicale (ad esempio, modelli che *imitano* le melodie tonali costruite nel sistema musicale tonale).
Un'impostazione semplicistica, ma che ben si presta per realizzare in breve esperienze musicali orientate alla sintesi automatica di melodie, consiste nel considerare la melodia come una successione di intervalli più o meno frequenti. Poiché la prassi musicale ci suggerisce la *bontà* o meno di una certa sequenza di intervalli, possiamo cercare di usare l'elaboratore e *istruirlo* sulla conoscenza di criteri per la composizione di melodie basati sulle successioni di intervalli. In altre parole, cerchiamo di mettere in grado l'elaboratore di generare automaticamente melodie sulla base di informazioni musicali riguardanti le sequenze di intervalli.
Come possiamo procedere? Ad esempio, possiamo associare a ogni sequenza costituita da due altezze (cioè a ogni intervallo) un valore di probabilità di *selezione*, cioè di *presenza media* di quell'intervallo nelle melodie generate; sulla base di queste informazioni costruiamo una tabella in cui associamo a ogni intervallo possibile (facendo riferimento alla scala temperata) un valore di probabilità che esprimiamo mediante l'ampiezza di un sottoinsieme dei numeri reali compresi tra 0 e 1. In questo modo, usando l'istruzione RND che genera, appunto, numeri reali pseudo-casuali compresi tra 0 e 1, associamo a ogni intervallo un sottoinsieme dei numeri reali compresi tra 0 e 1, proporzionalmente al valore di probabilità specificato per quell'intervallo. Un esempio di uso *bruto* dell'istruzione RND per la generazione pseudo-casuale di sequenze di note è dato dal seguente programma:

```
010 RN=RND(1)
020 X=RN*1e3+3
030 Y=RN*255
040 SOUND X,Y
050 GOTO 010
```

Tabella A: corrispondenze tra intervalli della scala temperata e subrange dell'intervallo 0-1 dei numeri reali per la realizzazione di una distribuzione di probabilità.

| intervallo (in semitoni) | intervallo | subrange associato | probabilità |
|---|---|---|---|
| −12 | ottava discendente | 0-0.03 | 3% |
| −11 | settima maggiore disc. | 0.03-0.04 | 1% |
| −10 | settima minore disc. | 0.04-0.06 | 2% |
| −9 | sesta maggiore disc. | 0.06-0.09 | 3% |
| −8 | sesta minore disc. | 0.09-0.12 | 3% |
| −7 | quinta disc. | 0.12-0.19 | 7% |
| −6 | quinta diminuita disc. | 0.19-0.20 | 1% |
| −5 | quarta disc. | 0.20-0.26 | 6% |
| −4 | terza maggiore disc. | 0.26-0.30 | 4% |
| −3 | terza minore asc. | 0.30-0.35 | 5% |
| −2 | seconda maggiore disc. | 0.35-0.41 | 6% |
| −1 | seconda minore disc. | 0.41-0.48 | 7% |
| 0 | unisono | 0.48-0.52 | 4% |
| +1 | seconda minore ascendente | 0.52-0.59 | 7% |
| +2 | seconda maggiore asc. | 0.59-0.65 | 6% |
| +3 | terza minore asc. | 0.65-0.70 | 5% |
| +4 | terza maggiore asc. | 0.70-0.74 | 4% |
| +5 | quarta asc. | 0.74-0.80 | 6% |
| +6 | quinta diminuita asc. | 0.80-0.81 | 1% |
| +7 | quinta asc. | 0.81-0.88 | 7% |
| +8 | sesta minore asc. | 0.88-0.91 | 3% |
| +9 | sesta maggiore asc. | 0.91-0.94 | 3% |
| +10 | settima minore asc. | 0.94-0.96 | 2% |
| +11 | settima maggiore asc. | 0.96-0.97 | 1% |
| +12 | ottava asc. | 0.97-1 | 3% |

Nella tabella A è riportata una sequenza di corrispondenze tra gli intervalli, i valori di probabilità associati e i sottoinsiemi dei reali associati (compresi tra 0 e 1); la scala di riferimento considerata è la scala temperata (le sequenze di altezze generate, cioè, saranno altezze appartenenti alla scala temperata). Gli intervalli considerati sono limitati agli intervalli, ascendenti e discendenti, fino a quello di ottava, per un'estensione complessiva di due ottave; in altre parole, non consideriamo intervalli superiori all'ottava, per brevità. In questa ipotesi, abbiamo intervalli più probabili (*quinta* e *seconda minore*) e intervalli meno probabili (*settima maggiore* e *quinta diminuita*); la distribuzione di probabilità è però simmetrica rispetto al verso ovvero la probabilità di un intervallo ascendente è uguale alla probabilità del corrispondente intervallo discendente.

## Un programma che genera melodie

Un programma che genera melodie secondo le informazioni della tabella A è riportato nella pagina a fronte.
Nel programma che abbiamo elencato, la tecnica di associare una certa probabilità ai parametri musicali è stata applicata anche al parametro delle durate sulla base delle corrispon-

## Un programma che genera melodie

```
001 REM INIZIALIZZAZIONE PARAMETRI
002 X=4697
003 Y=64
004 INPUT "SCRIVI UN NUMERO INTERO";W
005 INPUT "SCALA TEMPERATA O PER TONI INTERI? (1 o 2)";Z
006 IF Z=1 GOTO 10
007 IF Z=2 GOTO 1000
008 PRINT "RISPONDI 1 o 2"
009 GOTO 004
010 RN=RND(W)
020 REM ALTEZZE SCALA TEMPERATA
030 IF RN>0 and RN<=0.03 THEN X=X*2
040 IF RN>0.03 and RN<=0.04 THEN X=X*1.887749
050 IF RN>0.04 and RN<=0.06 THEN X=X*1.781797
060 IF RN>0.06 and RN<=0.09 THEN X=X*1.681793
070 IF RN>0.09 and RN<=0.12 THEN X=X*1.597401
080 IF RN>0.12 AND RN<=0.19 THEN X=X*1.498307
090 IF RN>0.19 AND RN<=0.20 THEN X=X*1.414214
100 IF RN>0.20 AND RN<=0.26 THEN X=X*1.334840
110 IF RN>0.26 AND RN<=0.30 THEN X=X*1.259921
120 IF RN>0.30 AND RN<=0.35 THEN X=X*1.189207
130 IF RN>0.35 AND RN<=0.41 THEN X=X*1.122462
140 IF RN>0.41 AND RN<=0.48 THEN X=X*1.059463
150 IF RN>0.52 AND RN<=0.59 THEN X=X/1.059463
160 IF RN>0.59 AND RN<=0.65 THEN X=X/1.122462
170 IF RN>0.65 AND RN<=0.70 THEN X=X/1.189207
180 IF RN>0.70 AND RN<=0.74 THEN X=X/1.259921
190 IF RN>0.74 AND RN<=0.80 THEN X=X/1.334840
200 IF RN>0.80 AND RN<=0.81 THEN X=X/1.414214
210 IF RN>0.81 AND RN<=0.88 THEN X=X/1.498307
220 IF RN>0.88 AND RN<=0.91 THEN X=X/1.597401
230 IF RN>0.91 AND RN<=0.94 THEN X=X/1.681793
240 IF RN>0.94 AND RN<=0.96 THEN X=X/1.781797
250 IF RN>0.96 AND RN<=0.97 THEN X=X/1.887749
260 IF RN>0.97 AND RN<=1.00 THEN X=X/2.0
270 GOTO 2800
1000 RN=RND(W)
1010 REM ALTEZZE SCALA PER TONI INTERI
1020 IF RN>0.00 and RN<=0.1 THEN X=X*1.122462
1030 IF RN>0.1 and RN<=0.2 THEN X=X/1.122462
1040 IF RN>0.2 and RN<=0.3 THEN X=X*1.259921
1050 IF RN>0.3 and RN<=0.4 THEN X=X/1.259921
1060 IF RN>0.4 and RN<=0.5 THEN X=X*1.414214
1070 IF RN>0.5 and RN<=0.6 THEN X=X/1.414214
1080 IF RN>0.6 and RN<=0.7 THEN X=X*1.597401
1090 IF RN>0.7 and RN<=0.8 THEN X=X/1.597401
1100 IF RN>0.8 and RN<=0.85 THEN X=X*1.781797
1110 IF RN>0.85 and RN<=0.9 THEN X=X/1.781797
2800 REM DURATE
2900 IF RN>0 and RN<=0.1 THEN Y=1
3000 IF RN>0.1 and RN<=0.2 THEN Y=2
3100 IF RN>0.2 and RN<=0.3 THEN Y=4
```

```
3200 IF RN>0.3 and RN<=0.4 THEN Y=8
3300 IF RN>0.4 and RN<=0.5 THEN Y=16
3400 IF RN>0.5 and RN<=0.6 THEN Y=32
3500 IF RN>0.6 and RN<=0.7 THEN Y=64
3600 IF RN>0.7 and RN<=0.8 THEN Y=128
3700 IF RN>0.8 and RN<=0.9 THEN Y=Y/2
3800 IF RN>0.9 and RN<=1.0 THEN Y=Y*2
5000 REM CONTROLLI
5100 IF X>16383 THEN X=X/2
5150 IF X<310 THEN X=X*2
5200 IF Y>255 OR Y<1 THEN Y=64
5300 SOUND X,Y
5400 GOTO 006
```

denze che vengono indicate nella tabella B.
Nel programma abbiamo anche una parte dedicata alla generazione di sequenze di altezze, in cui si fa riferimento alla scala per toni interi e alla tabella C per la distribuzione di probabilità.
Per risparmiare tempo di calcolo abbiamo precalcolato le radici e le potenze di 2: in questo modo abbiamo tempi di attesa inferiori tra una nota e la successiva.
A completamento della trattazione del parametro altezza, ricordiamo che un suono musicalmente molto importante è il *silenzio*! In musica, il silenzio è descritto mediante la *pausa*; in corrispondenza, per ogni *figura* di durata (croma, biscroma ecc.) usiamo un segno specifico come è mostrato nelle illustrazioni qui sotto. Per ottenere una pausa con l'istruzione **SOUND** basterà specificare un valore di frequenza non percepibile, per esempio:

SOUND 0, Y

**Tabella B: corrispondenze tra valori del parametro y (durata) e subrange dell'intervallo 0-1 dei numeri reali per la realizzazione di una distribuzione di probabilità.**

| valore del parametro Y | subrange | probabilità |
|---|---|---|
| 1 | 0-0.1 | 10% |
| 2 | 0.1-0.2 | 10% |
| 4 | 0.2-0.3 | 10% |
| 8 | 0.3-0.4 | 10% |
| 16 | 0.4-0.5 | 10% |
| 32 | 0.5-0.6 | 10% |
| 64 | 0.6-0.7 | 10% |
| 128 | 0.7-0.8 | 10% |
| Y/2 (dimezzamento della figura) | 0.8-0.9 | 10% |
| Y*2 (raddoppio della figura) | 0.9-1 | 10% |

**Tabella C: corrispondenze tra intervalli della scala per toni interi e subrange dell'intervallo 0-1 dei numeri reali per la realizzazione di una distribuzione di probabilità.**

| intervallo (in toni interi) | subrange associato | probabilità |
|---|---|---|
| −1 (un tono discendente) | 0-0.1 | 10% |
| +1 (un tono ascendente) | 0.1-0.2 | 10% |
| −2 (due toni disc.) | 0.2-0.3 | 10% |
| +2 (due toni asc.) | 0.3-0.4 | 10% |
| −3 (tre toni disc.) | 0.4-0.5 | 10% |
| +3 (tre toni asc.) | 0.5-0.6 | 10% |
| −4 (quattro toni disc.) | 0.6-0.7 | 10% |
| +4 (quattro toni asc.) | 0.7-0.8 | 10% |
| −5 (cinque toni disc.) | 0.8-0.85 | 5% |
| +5 (cinque toni asc.) | 0.85-0.9 | 5% |
| unisono | 0.9-1 | 10% |

*Lezione 6*

## Ancora sulle iterazioni

Immaginiamo di scrivere una parte di programma che acquisisce i comandi dall'utente per selezionare una tra le funzioni offerte da un menù.
Si pensi ad esempio a un programma che consente di calcolare il volume di diversi solidi (sfera, cubo, piramide, cilindro...).
Un tale programma può iniziare visualizzando il "menù" delle scelte possibili nel seguente modo:

Calcolo Volume:
1. Sfera
2. Cubo
3. Piramide
4. Cilindro
0. Fine programma

Numero funzione selezionata?

Si richiede che il programma continui a riproporre il menù e a effettuare il calcolo richiesto, fino a quando l'utente non seleziona il comando di terminazione, fornendo il valore 0.
Il programma che realizza la suddetta gestione del menù si presenta così in PASCAL:

```
acquisisci comando
WHILE comando ‹›0 DO
   esegui comando
   acquisisci comando
```

Il programma presentato si comporta così:

- acquisisce un primo comando
- se il comando non è 0

allora
esegue il comando
acquisisce un nuovo comando
torna ad eseguire il controllo di comando =0

## Caratteristiche della struttura "WHILE...DO"

Anche questa è dunque una struttura di controllo iterativa: perché non utilizzare la REPEAT...UNTIL già vista?

*Completata questa sesta lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi*
*RADQT.DO*
*RADQP.BA*
*contenuti nella cassetta "11 Esercizi di Programmazione".*
*I titoli seguiti dal suffisso DO corrispondono a testi, quelli seguiti da BA a programmi in BASIC.*
*Caricateli secondo le modalità che avete appreso.*

Osserviamo alcune caratteristiche della struttura precedentemente mostrata:

- il controllo sul valore del comando acquisito viene effettuato appena si entra nella struttura: ciò comporta che, se la condizione verificata risulta falsa (è stato cioè fornito il comando 0), l'iterazione non viene eseguita nemmeno una volta: questa è una prima differenza rispetto alla struttura REPEAT...UNTIL; quest'ultima infatti effettua il controllo alla fine dell'iterazione, che pertanto verrà comunque eseguita almeno una volta;
- l'iterazione viene eseguita quando la condizione espressa (nel nostro caso "comando ‹› 0") risulta vera; pertanto la condizione che determina l'uscita dalla iterazione è la negazione della condizione espressa (nel nostro caso "comando = 0"); questa è un'altra differenza rispetto alla struttura REPEAT...UNTIL, che prevede l'esecuzione del blocco iterativo se la condizione espressa è falsa e quindi l'interruzione dell'iterazione quando la condizione risulta vera.

Quando usare dunque la struttura REPEAT...UNTIL e quando la WHILE...DO?
Si tratta essenzialmente di sfruttare opportunamente le caratteristiche delle due strutture in rapporto alle esigenze dell'algoritmo e della chiarezza del programma.
Per esempio, nel caso del programma che visualizza i primi cento numeri è più opportuno l'uso della REPEAT...UNTIL: infatti è inutile controllare il valore del contatore fin dalla prima passata, poiché comunque è certo che le istruzioni del blocco iterativo devono essere eseguite.
Nel caso invece del programma che gestisce i comandi forniti da un utente, è importante che il controllo sulla condizione sia effettuato subito: l'utente, infatti, potrebbe accorgersi di aver sbagliato e voler quindi interrompere il programma senza eseguire nemmeno una volta l'iterazione.

## La struttura WHILE...DO in BASIC

Esaminiamo adesso la versione BASIC del programma precedentemente visto:

*N.B. Il ● sta a significare che la linea va a capo per esigenze editoriali, quindi nell'eseguirla non interrompere la digitazione.*

```
10 PRINT "CALCOLO VOLUME:"
20 PRINT "    1. SFERA"
30 PRINT "    2. CUBO"
40 PRINT "    3. PIRAMIDE"
50 PRINT "    4. CILINDRO"
60 PRINT "    0. FINE PROGRAMMA"
70 PRINT
80 INPUT "Numero funzione selezionata":C
90 IF C=0 THEN 200
100 REM Inserire qui l'analisi della selezione●
e relativo calcolo
170 REM Nuova richiesta di selezione
180 INPUT "Numero funzione selezionata":C
190 GOTO 90
200 REM Fine iterazione
```

Anche in questo caso non è disponibile una struttura di controllo *ad hoc* e pertanto la "WHILE...DO" è stata realizzata con le istruzioni BASIC già viste.
Si tratta quindi di effettuare il controllo all'inizio dell'iterazione e di esprimere cor-

rettamente la condizione nell'istruzione IF.
Si noti come l'istruzione PRINT usata senza argomenti permetta di ottenere un'interlinea sul video.

### La programmazione strutturata

Nel 1960, il professor E.W. Dijkstra, autorevole ricercatore olandese, inviava una lettera alla prestigiosa rivista di informatica *COMMUNICATION OF ACM*, sostenendo che, nella sua esperienza, la qualità di un programmatore era tanto migliore quante meno istruzioni GOTO questo usava.
Ne nasceva una diatriba durata non poco tempo, che riconosceva in quell'osservazione il fatto che spesso programmi ricchi di istruzioni GOTO corrispondono a programmi "poco pensati" e che indicava come soluzione metodologica al problema l'uso della PROGRAMMAZIONE STRUTTURATA.
Con tale termine si intende il modo di procedere a cui il nostro testo si è attenuto, che prevede:

- l'uso di tre sole strutture di controllo a un ingresso e a un'uscita:
  - SEQUENZA
  - SELEZIONE
  - ITERAZIONE
- lo sviluppo top down.

L'adozione di tali strumenti permette infatti di pensare di più quando si costruiscono programmi, e quindi di ottenere programmi con meno errori, più facili da comprendere, più aderenti al problema da risolvere; mentre, al contrario, il mancato rispetto di tali regole porta spesso (in particolare sui programmi di dimensioni medie o grandi) a quello che gli americani chiamano *spaghetti programs*, con riferimento all'intricatezza di tutti i salti che non permettono di venire a capo delle idee che stanno alla base del programma.
In seguito, la disciplina si è arricchita di numerose considerazioni sul modo di esaminare i dati del problema e quindi di organizzare le variabili dei programmi, che avremo modo di vedere con dettaglio.
Attualmente il termine PROGRAMMAZIONE STRUTTURATA è poco usato, poiché è universalmente dato per scontato che non esiste un ragionevole modo di procedere differente da quello, per cui si parla semplicemente di "programmazione".

## Un esempio

Supponiamo di volere calcolare la somma di un certo insieme di valori, il cui numero non è noto a priori: possiamo costruire un programma che legge uno per uno i valori e li totalizza successivamente, fino al momento in cui non viene letto un valore speciale "proibito" da considerare come indicazione di "fine dati".
Ad esempio, se tutti i valori da sommare sono positivi, l'indicatore di fine può essere –1.
Il programma può essere realizzato in due modi differenti, usando rispettivamente le strutture REPEAT...UNTIL e WHILE...DO, secondo i modelli:

```
S:=0
leggi X
REPEAT
   S:=S+X
   leggi X
UNTIL X=-1
visualizza S
```

oppure

```
S:=0
leggi X
WHILE X ‹› -1 DO
   BEGIN
      S:=S+X
      leggi X
   END
visualizza X
```

(si noti la coppia BEGIN-END, che identifica tutte le istruzioni che devono essere iterate dalla struttura WHILE!)

In BASIC, la prima struttura risulta:

```
10 LET S=0
20 INPUT X
20 REM inizio repeat
50 LET S=S+X
55 INPUT X
60 IF X<>-1 THEN 30
70 PRINT "S=";S
```

mentre la seconda risulta:

```
10 LET S=0
20 INPUT X
30 REM inizio WHILE
40 IF X=-1 THEN GOTO 70
50 LET S=S+X
60 INPUT X
65 GOTO 40
70 PRINT "S=";S
```

Provate a eseguire i due programmi: il loro funzionamento è analogo se c'è almeno un dato da sommare, ma se forniamo come elemento iniziale il valore –1 per dire che non c'è alcun dato, il primo programma non è adeguato.

### Cosa abbiamo imparato

In questa lezione abbiamo visto:

- il concetto di "menù", una tecnica molto utile per rendere di facile uso programmi che offrono più funzioni;
- la struttura iterativa WHILE...DO
- le differenze tra WHILE...DO e REPEAT...UNTIL
- che cos'è la PROGRAMMAZIONE STRUTTURATA.

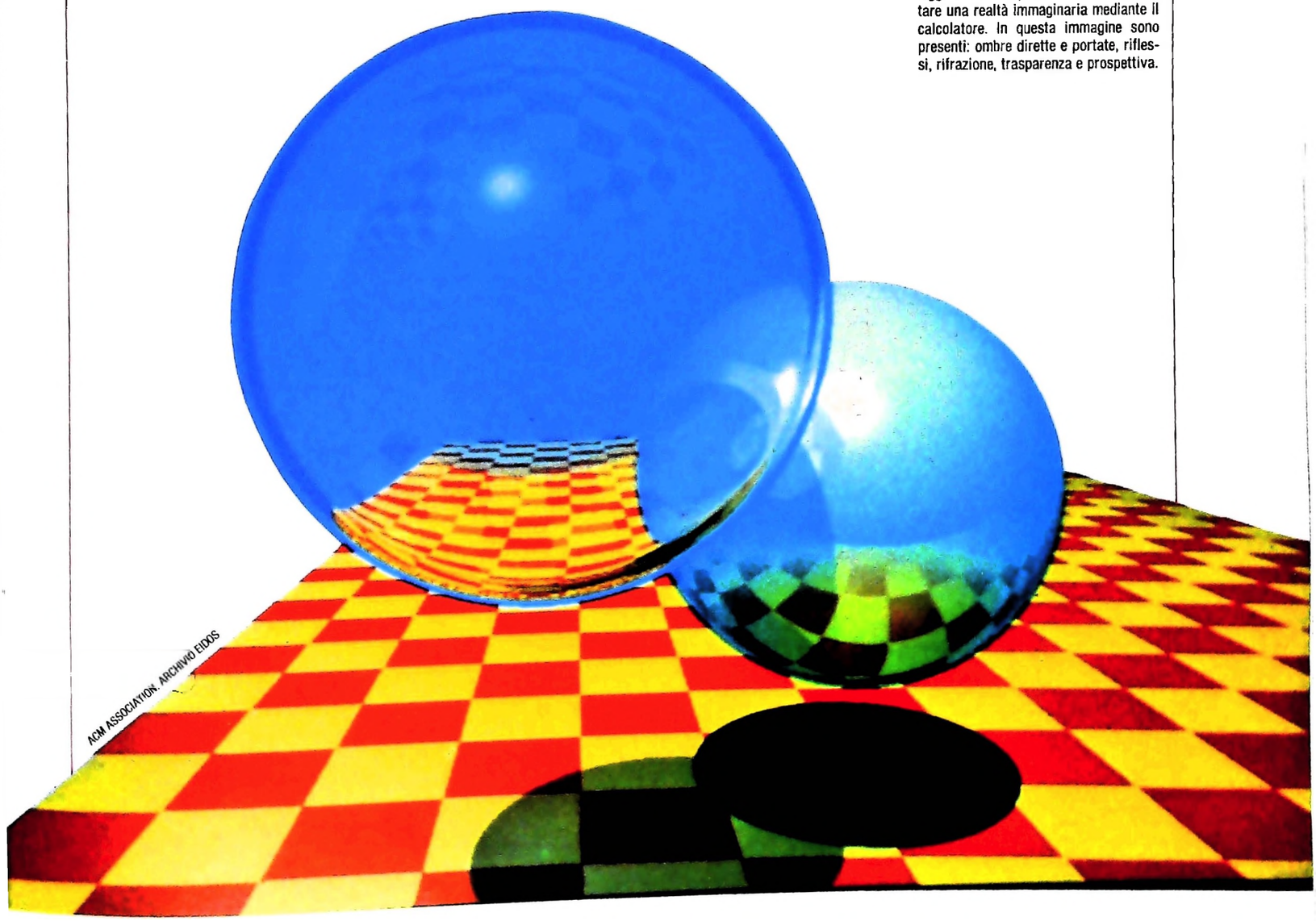
# DISEGNARE UN CERCHIO

## Impariamo come sia possibile visualizzare un cerchio sul display di M10.

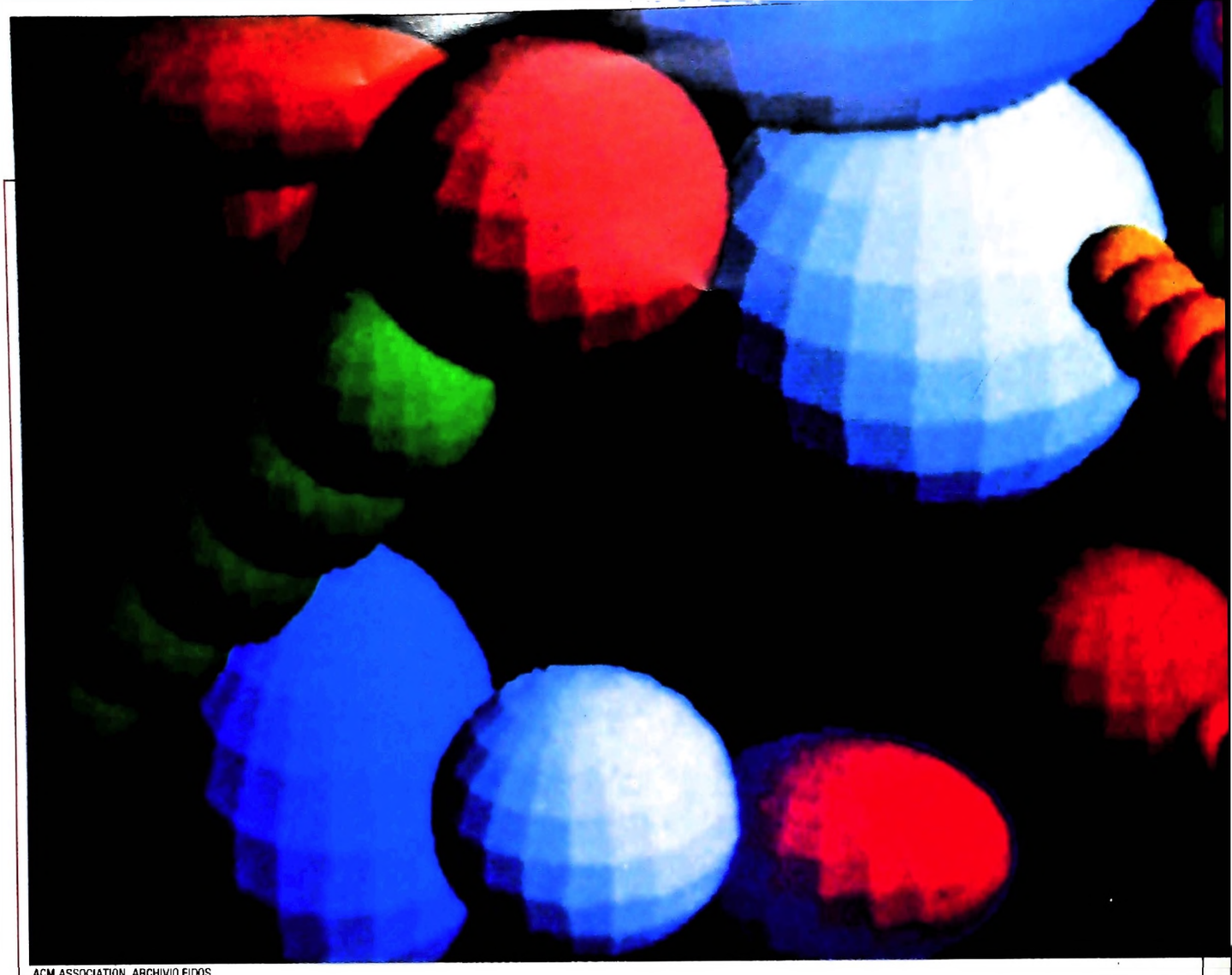
Affrontiamo il problema di disegnare in un modo molto semplice la più regolare fra tutte le figure geometriche: il cerchio.
Per questo dobbiamo anzitutto avere ben presente la risoluzione grafica che ci mette a disposizione lo schermo del nostro computer. Esso è costituito, nel caso dell'M10, da una griglia di 64 per 240 pixel.

Di conseguenza dobbiamo necessariamente adeguarci all'approssimazione grafica che può essere ottenuta con una tale risoluzione.
Per renderci conto di come agisce la griglia di pixel, immaginiamo di dover disegnare un cerchio su un foglio a quadretti. Supponiamo che il raggio sia di 4 quadretti e disegnamolo con un compasso. Ora anneriamo tutti i quadrettini che sono

Un accurato modello matematico delle leggi dell'ottica permette di rappresentare una realtà immaginaria mediante il calcolatore. In questa immagine sono presenti: ombre dirette e portate, riflessi, rifrazione, trasparenza e prospettiva.

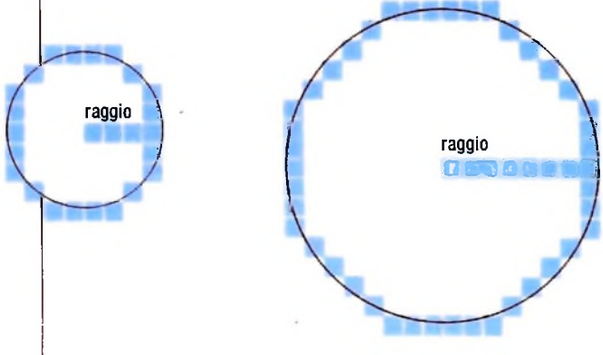


ACM ASSOCIATION. ARCHIVIO EIDOS

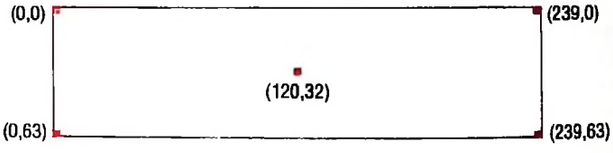


ACM ASSOCIATION, ARCHIVIO EIDOS

toccati dalla circonferenza, e proviamo ad immaginare come verrebbero illuminati i pixel sullo schermo dell'M10 per disegnare lo stesso cerchio.
Se ora pensiamo di disegnare un cerchio con il raggio di 8 quadretti, potremo seguire la stessa procedura, arrivando a ottenere l'immagine qui sotto.



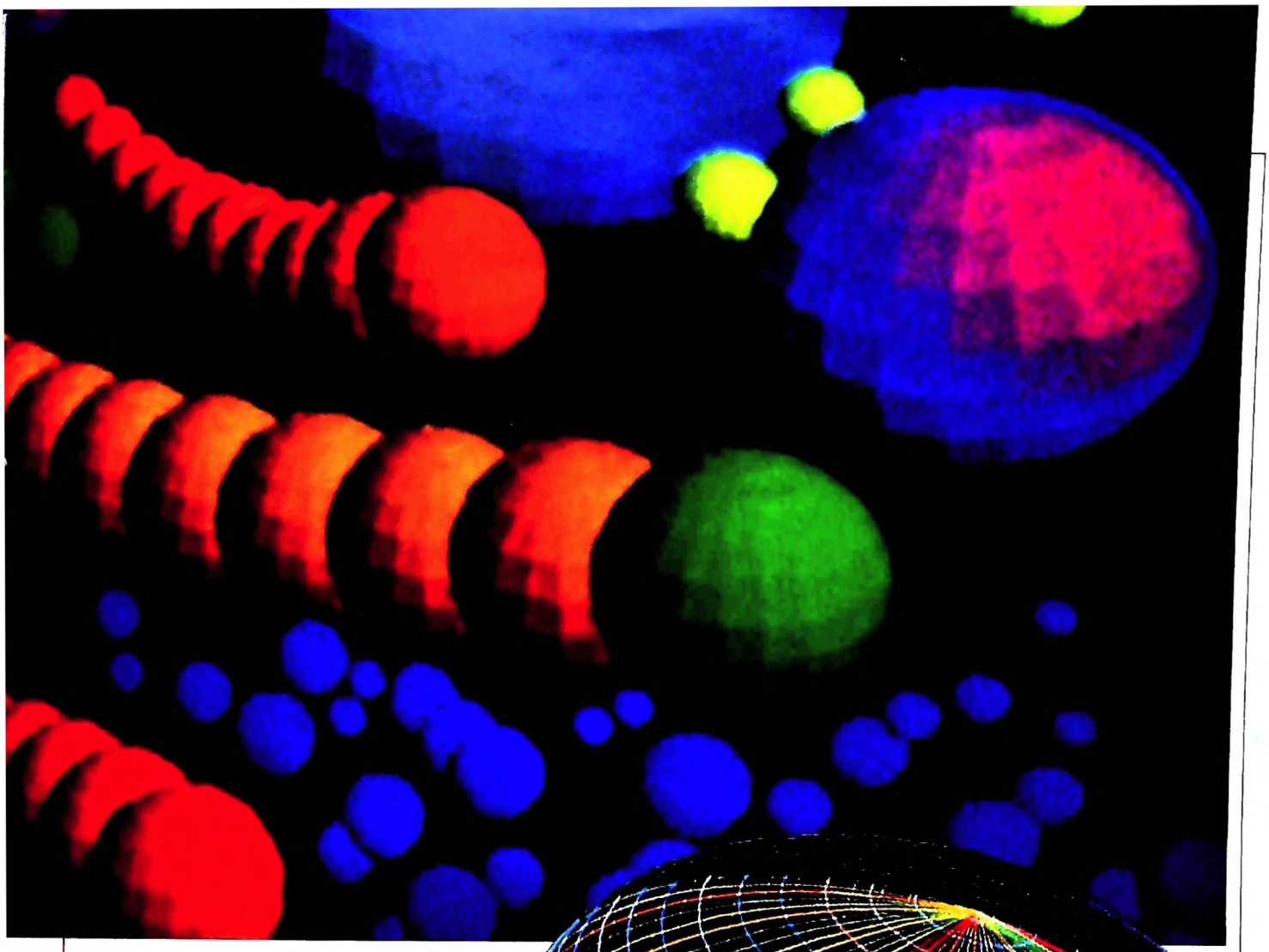

Questo esempio illustra come la precisione con cui sullo schermo viene disegnato un cerchio dipende dalla lunghezza del raggio.
Vediamo ora di capire come si deve fare per determinare i pixel che devono essere illuminati nel disegno di un cerchio. Osserviamo innanzitutto che un qualsiasi pixel dello schermo è identificabile mediante una coppia di numeri in maniera ovvia: ossia, stabiliamo che i pixel posti ai quattro vertici del nostro schermo siano identificati dalle coppie (0, 0), (0, 63), (239, 63) e (239, 0).

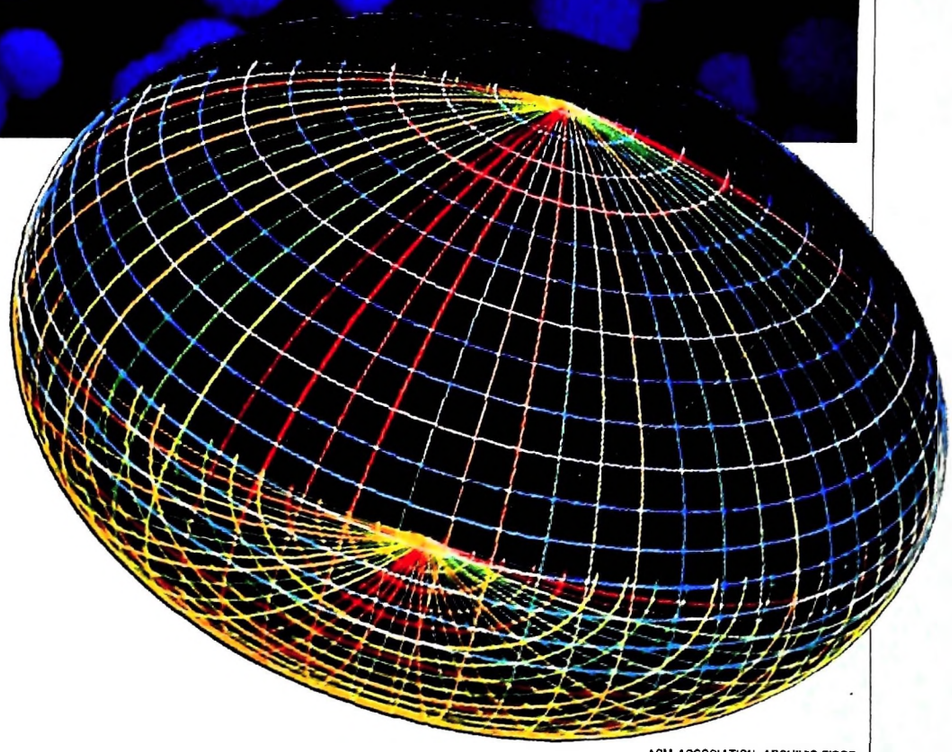



Un qualsiasi pixel P dello schermo viene allora identificato con una coppia di numeri (x, y), che chiamiamo coordinate, in cui x e y indicano rispettivamente di quanti pixel bisogna spostarsi orizzontalmente e verticalmente, a partire dal pixel (0, 0), per trovare P. Per esempio, il punto (120, 32) rappresenta il pixel che sta nel centro dello schermo.
Ritorniamo ora al problema del cerchio. La geometria analitica ci insegna che le coordinate dei punti che costituiscono una circonferenza di raggio R verificano una precisa legge matematica, che può essere espressa mediante queste due equazioni:

$$x = R \cos(ang)$$
$$y = R \sin(ang)$$

in cui R è il raggio del cerchio e "ang" è l'angolo di cui si





Un mondo di sfere viene approssimato con dei polledri regolari a facce piane, colorate con tinte di intensità cromatiche diverse; si genera così un effetto tridimensionale e si simula l'esistenza di una sorgente luminosa direzionale (qui sopra). Ruotando di 360° un arco di cerchio come sul tornio di un vasaio, si può ottenere una forma regolare come quella qui accanto.

ACM ASSOCIATION, ARCHIVIO EIDOS

ruota il raggio fra un pixel e il successivo. Bisogna ora specificare una caratteristica dell'M10. Esso accetta per gli angoli solo valori espressi in radianti e non in gradi sessagesimali. Si ricorda che l'angolo di 360 gradi vale $2\pi$ radianti, e quindi ogni altro angolo è ottenibile come frazione di $2\pi$: per esempio, l'angolo di 36 gradi è uguale a $360/10 = 2\pi/10 = \pi/5$.

Possiamo ora fare una considerazione: pensiamo a come fa un giardiniere a disegnare un'aiuola circolare. Prende una corda, ne fissa un estremo nel terreno e procede a farla ruotare di $2\pi$ radianti, cioè di 360 gradi, tenendola sempre tesa. Allora, sul video dell'M10, l'insieme dei pixel che calpesterebbe il nostro giardiniere girando con la corda tesa è un cer-

chio, il cui raggio risulta pari alla lunghezza della corda.
Tale cerchio sarà sempre costituito da un numero finito di pixel, che varieranno in funzione dell'ampiezza dell'angolo percorso dal raggio fra un pixel e'il successivo. In altre parole, se vogliamo una circonferenza costituita da 36 punti dovremo calcolare 36 volte i valori di x e y in base alle due equazioni date e ogni volta il valore della variabile "ang" dovrà essere incrementato di $\pi/18$ radianti, cioè 10 gradi, a partire dal valore 0: se vorremo invece una circonferenza costituita da 72 punti dovremo dare per 72 volte l'incremento di $\pi/36$ radianti, cioè 5 gradi.

Si può quindi affermare che, posto N uguale al numero dei pixel che descrivono il cerchio, l'incremento che si deve attribuire alla variabile "ang" a ogni passaggio sarà dato dall'equazione:

$$ang = 2\pi/N$$

oppure, viceversa, fissato l'incremento si può sapere di quanti pixel sarà costituito il cerchio risolvendo l'equazione:

$$N = 2\pi/ang$$

## Un programma per disegnare cerchi

Presentiamo qui un semplice programma in BASIC che consente di realizzare il disegno di un cerchio sullo schermo di M10. Il cerchio disegnato è formato da 36 punti; discuteremo poi come si possa modificare il programma per cambiare tale valore.

```
10  INPUT "Quanto e' lungo il raggio "; R
20  PIGRECA=3.1415
30  INPUT "Quali devono essere le coordinate del cen●
tro "; XCENTRO,YCENTRO
40  PRINT "Per avere il disegno del cerchio batti ENTER!";ENTER
50  CLS
60  FOR ANG = 0 TO 2*PIGRECA  STEP PIGRECA/18
70  X = XCENTRO + R*COS(ANG)
80  Y = YCENTRO + R*SIN(ANG)
90  PRESET (X,Y,0)
100 NEXT ANG
110 END
```

*N.B. Il ● sta a significare che la linea va a capo per esigenze editoriali, quindi nell'eseguirla non interrompere la digitazione.*

## Commento al programma

La prima istruzione, che ha numero 10, ci chiede di fornire un numero che rappresenterà la lunghezza in pixel del raggio. Nell'istruzione 20 viene definito il valore della costante PIGRECA. Nella 30 si chiede di definire le coordinate del centro del cerchio, cioè di definire un pixel che costituirà il centro della circonferenza. La 40 ci richiede di battere ENTER per ottenere il disegno del cerchio. La 50 serve per ripulire lo schermo dalle scritte precedenti. Le istruzioni 60 e 100 costituiscono un cosiddetto "ciclo di FOR". Viene qui definita una variabile ANG che può assumere valori da 0 a 2 PIGRECA, con incrementi di PIGRECA/18, cioè di 10 gradi, per volta. Questo significa che la prima volta la variabile ANG vale 0 e vengono eseguite le istruzioni 70, 80 e 90 con questo valore per tale variabile. L'istruzione 100 incrementa automaticamente di PIGRECA/18 il valore di ANG e vengono di nuovo eseguite le istruzioni 70, 80 e 90 con tale nuovo valore. Il ciclo continua finché ANG vale 2 PIGRECA; dopo di che il programma si conclude.

Il programma disegna un cerchio di 36 punti. Potremmo però voler decidere di volta in volta, usando lo stesso programma, il numero di punti che devono costituire il cerchio. Per fare ciò dovremo apportare qualche modifica al programma, e cioè fornire il numero di punti desiderato e inserire una istruzione che calcoli, in base a questo numero, l'incremento da assegnare alla variabile ANG. Bisognerà quindi aggiungere le seguenti istruzioni:

```
25  INPUT "Quanti punti deve ave●
re il cerchio?"; N
26  I = 2*PIGRECA / N
```

Ovviamente ci sarà anche da variare l'istruzione 60 nel seguente modo:

```
60 FOR ang = 0 to 2*PIGRECA  STEP I
```

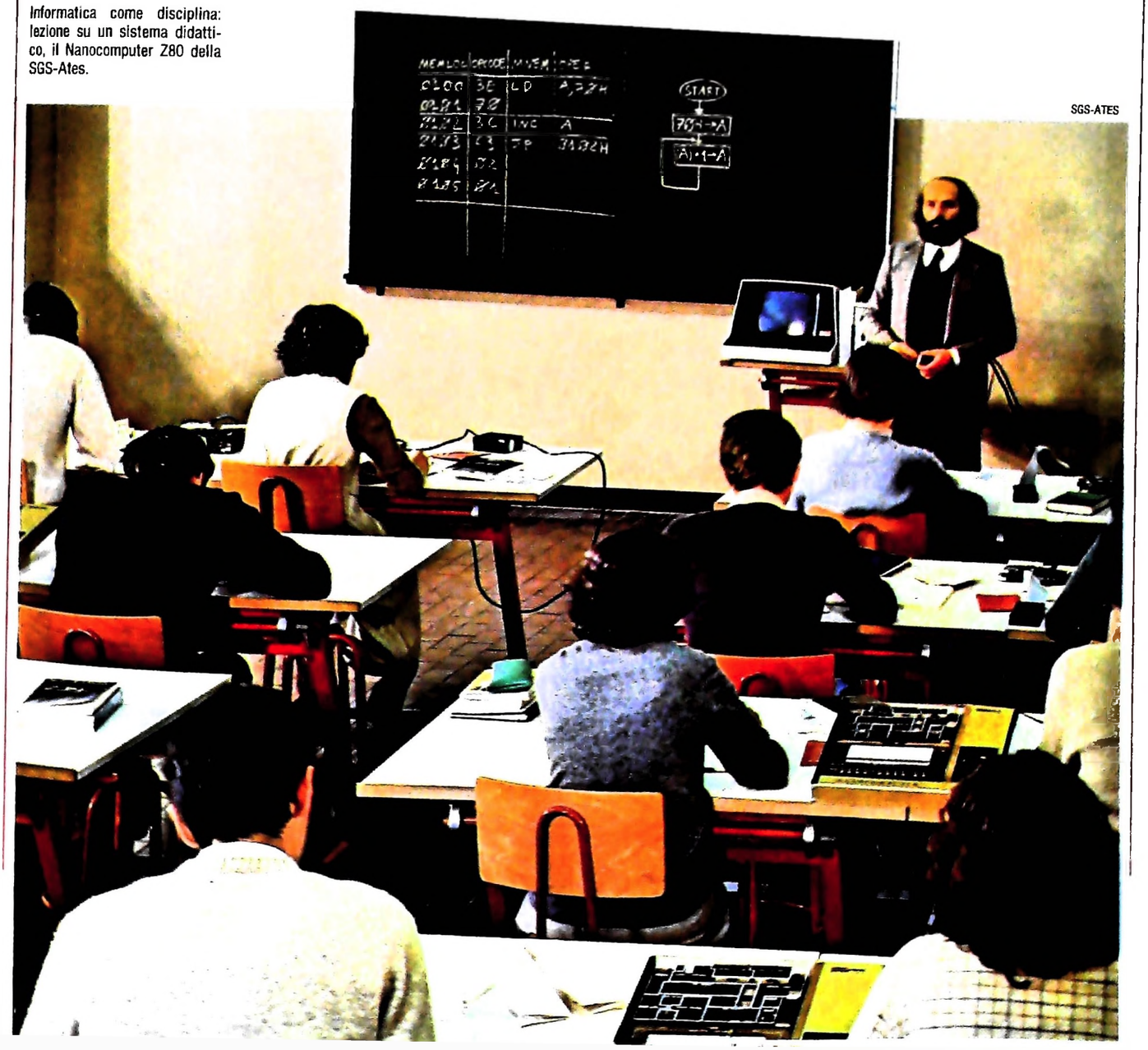
# Algoritmi e strutture

## Gli aspetti formativi nello studio e nella pratica dell'informatica.

Quando si parla di informatica e didattica, occorre distinguere due aspetti fondamentali: l'informatica come disciplina oggetto d'insegnamento e l'informatica come strumento per l'insegnamento di altre discipline. Due momenti nettamente distinti anche se in continua interazione, ciascuno dei quali ha caratteristiche, ambiti e problemi propri.
Possiamo pensare anche a una sintesi tra di essi ed è quando agli allievi si insegna a preparare programmi per uso didattico, che cioè dovranno essere impiegati in modi diversi per l'insegnamento di altre materie. Questa sintesi, che noi riteniamo ideale, non è però tanto facile da raggiungere perché i programmi per uso didattico in genere sono complessi e non facilmente realizzabili nei normali corsi di informatica a livello più elementare.
Quale che sia il livello di insegnamento dell'informatica, esiste comunque un vantaggio fondamentale ed è il valore formativo legato a questa disciplina. Più specificamente, cogliamo tre aspetti formativi nello studio e nella pratica dell'informatica, e cioè l'impostazione algoritmica, la decomposizione gerarchica e l'astrazione funzionale.

Informatica come disciplina: lezione su un sistema didattico, il Nanocomputer Z80 della SGS-Ates.

SGS-ATES



## L'impostazione algoritmica

Algoritmo è una qualunque procedura, cioè una sequenza di operazioni che conduce al raggiungimento di un certo risultato, solitamente alla soluzione di un problema.
L'algoritmo non è nato certamente con l'informatica; possiamo anzi trarre dalla vita quotidiana stessa esempi di algoritmo in senso generale. Pensiamo alle procedure per eseguire qualunque operazione, dal cuocere un uovo al tegamino al cambiare una ruota d'automobile, dal fare una telefonata al lavorare all'uncinetto ecc. Nella matematica poi, per essere più vicini al campo che ci interessa, gli algoritmi non si contano.
Eppure l'informatica ha un merito speciale: ha costretto alla formulazione precisa di algoritmi, per la necessità di tradurli poi in un linguaggio perfettamente accettabile da una macchina.
L'atteggiamento algoritmico richiede la capacità di elaborare modelli di sistemi, di manipolare e modificare tali modelli per approssimare il comportamento dei sistemi reali, di passare continuamente dal sistema al modello (e viceversa) per controllarne la corrispondenza reciproca e verificare l'adeguatezza del secondo al primo.

## La decomposizione gerarchica

La decomposizione gerarchica si potrebbe definire sinteticamente una tecnica di passaggio dal complesso al semplice, un modo cioè di dominare concettualmente, e quindi di manipolare, una realtà complessa.
Tecnicamente il procedimento si chiama *top-down*, cioè letteralmente "dall'alto in giù" e consiste nel suddividere un problema in elementi fondamentali, ciascuno dei quali a sua volta viene suddiviso in sottoproblemi a un livello più basso, e così via fino alla massima semplificazione possibile o comunque compatibile con il problema in questione e con i mezzi a disposizione per risolverlo.
Anche in questo caso, chiaramente non si tratta di un atteggiamento soltanto informatico, ma l'informatica, specialmente per quanto riguarda la programmazione strutturata, di cui parleremo tra breve, ha costretto ad adottarlo in modo sistematico e preciso.

## L'astrazione funzionale

Una elaborazione nel suo complesso può essere vista come una funzione astratta che opera sui dati (ingresso) e produce risultati (uscita). Associazione logica, quindi, tra dati e risultati, mediata dall'algoritmo, tradotta in un linguaggio di programmazione e resa operante dalla macchina che esegue l'operazione.
L'astrazione funzionale corrisponde a una capacità di base ancora più a monte dello stesso atteggiamento algoritmico e si colloca addirittura tra le operazioni legate al conoscere stesso. Conoscere infatti significa identificare e discriminare: identificare un individuo come tale oppure come appartenente a una determinata classe, e discriminarlo da tutto ciò che non è e da tutto ciò a cui non appartiene.
L'identificazione e la discriminazione si attuano attraverso un complesso di indizi, di segni, di collegamenti. Più esattamente: di correlazioni. E la funzione matematica rappresenta lo strumento tipico della correlazione, secondo le regole schematizzate a suo tempo da Francesco Bacone.
L'informatica ha "costretto" chi vi lavora ad assumere questi atteggiamenti, a precisarli, a svilupparli fino a farli diventare abito mentale.
Il tutto si è ulteriormente sviluppato e precisato attraverso uno strumento di acquisizione relativamente recente e cioè la programmazione strutturata.

## La programmazione strutturata

Con questo termine si intende un metodo ormai consolidato nel campo dell'informatica e la cui efficacia è fuori discussione; costituisce infatti la risposta più avanzata alla principale esigenza della programmazione, e cioè la razionalizzazione dei procedimenti.
Riassumiamo i risultati ottenibili da questo metodo:
a) miglioramento della didattica della programmazione, che è uno dei momenti fondamentali dell'informatica;
b) facilitazione della lettura e dell'interpretazione di un programma da parte di persone diverse dall'autore, o anche da parte dell'autore stesso in tempi successivi alla prima stesura del programma stesso;
c) semplificazione dei procedimenti di ricerca degli errori in un programma e in generale di modifiche dello stesso, sia da parte dell'autore sia da parte di altri.
Gli strumenti della programmazione strutturata sono due: le strutture di controllo sono ridotte a tre tipi fondamentali, l'analisi del problema e il relativo passaggio all'algoritmo risolutivo avvengono mediante il procedimento *top-down*.
Ogni programma consiste essenzialmente di due elementi: dati e istruzioni che operano su di essi. A loro volta le istruzioni sono di due tipi: quelle che operano sui dati immediatamente, ad esempio gli operatori aritmetici o in genere funzionali, e gli operatori che organizzano i precedenti, cioè che controllano in generale l'esecuzione del programma, stabilendo l'ordine di svolgimento delle singole istruzioni, le precedenze, i salti ecc.
Le istruzioni di questo secondo tipo si definiscono "strutture di controllo". Ebbene, nella programmazione strutturata le strutture di controllo sono solo di tre tipi, e precisamente la sequenza, la selezione e l'iterazione.
La *sequenza* è il tipo più semplice: le istruzioni elementari vengono eseguite una dopo l'altra, nello stesso ordine in cui compaiono nel programma.
Nel caso della *selezione* si hanno due blocchi (sequenze) di istruzioni: a seconda del verificarsi o meno di una data condizione viene eseguita la prima o la seconda sequenza. Si tratta di una scelta condizionata.
L'*iterazione* rappresenta la ripetizione ciclica di una sequen-

za di istruzioni: le stesse istruzioni operano a ogni ciclo su dati diversi e l'intera operazione si conclude quando scatta una opportuna condizione.
La concatenazione tra le diverse strutture di controllo si effettua attraverso il procedimento *top-down*, di cui si è appena parlato a proposito del valore formativo della decomposizione gerarchica.
Si tratta di descrivere il procedimento da tradurre in programma, a partire dal livello più alto; di sviluppare successivamente i vari elementi contenuti in questo livello, scendendo così al secondo livello; di sviluppare ulteriormente gli elementi di questo secondo livello e di "scendere" successivamente di livello in livello fino a che il procedimento risolutivo è descritto in modo preciso ed esauriente.
La descrizione viene contemporaneamente tradotta, livello per livello, in una delle tre strutture fondamentali di controllo. A titolo d'esempio presentiamo il programma di un gioco, il "gioco del 15".

## Gioco del 15 - algoritmo

```
INIZIO
    Visualizzazione regole del gioco
    SE Avversario sceglie la seconda mossa (il calcolatore
                    gioca per primo)
       ALLORA
          Scegli N=3
          M=3
          ESEGUI MENTREVAL M<15
             Aspetta mossa avversario, che gioca N
             M=M+N
             Scegli 4-N
             M=M+4-N
          RIPETI
          Stampa "Ho VINTO!"
       ALTRIMENTI
          ESEGUI MENTREVAL M<15
             Aspetta mossa avversario, che gioca N
             SE N=1
                ALLORA
                   M=M+1
                   Scegli 2
                   M=M+2
                   ESEGUI MENTREVAL M<15
                      Aspetta mossa avversario, che gioca N
                      M=M+N
                      Scegli 4-N
                      M=M+4-N
                   RIPETI
                   Stampa "Ho VINTO!"
             OPPURESE N=2
                ALLORA
                   M=M+2
                   Scegli 1
                   M=M+1
                   ESEGUI MENTREVAL M<15
                      Aspetta mossa avversario, che gioca N
                      M=M+N
                      Scegli 4-N
                      M=M+4-N
                   RIPETI
                   Stampa "Ho VINTO!"
                ALTRIMENTI
                   M=M+3
                   Scegli 1
                   M=M+1
             FINESE
          RIPETI
          Stampa "Ho Perso"
       FINESE
FINE
```

### La strategia vincente

Due giocatori a turno scelgono un numero intero tra 1 e 3; ogni numero si somma al totale precedente; vince chi arriva al numero 15. La strategia vincente consiste nel raggiungere uno qualunque dei seguenti numeri: 3, 7, 11, e quindi nel giocare la differenza tra il numero 4 e quello giocato dall'avversario. Il programma è predisposto con la strategia ottimale e cioè se l'elaboratore gioca per primo allora gioca il numero 3, seguito dalla differenza tra il 4 e il numero dell'avversario. Se gioca per secondo, cerca di raggiungere uno dei numeri chiave non appena possibile, per poi applicare la strategia vincente. Qui, la struttura del programma con la tecnica della programmazione strutturata. La selezione si ottiene attraverso le quattro parole chiave SE, ALLORA, ALTRIMENTI, FINESE. Se le scelte sono più di due si aggiunge una quinta parola e cioè OPPURESE. L'iterazione si realizza attraverso tre parole chiave: ESEGUI MENTREVAL (cioè finché vale la condizione seguente) e RIPETI, che chiude la sequenza che viene iterata. L'algoritmo traduce la strategia ottimale, probabilmente vincente, se l'avversario non la conosce. I vari livelli logici si differenziano con i diversi livelli di rientro delle righe.

## Gioco del 15 - programma per M10

```
10 REM Gioco del 15
20 CLS
30 PRINT "Ogni giocatore sceglie a turno un numero 1, 2 o 3 / Ogni numero si somma ai     precedenti"
40 PRINT "Vince il giocatore che raggiunge il 15"
50 PRINT "Vuoi giocare per primo? Imposta 1.Al-"
60 PRINT "trimenti imposta qualunque altro numero"
70 INPUT A
80 CLS
90 IF A=1 THEN 250
100 '-------------------------------------------------------------------------------------------------------
110 N=3
120 M=3
130 PRINT "Ho scelto 3. Adesso scegli tu"
140 INPUT N
150 IF FIX(N))(N OR N)3 OR N(1 THEN PRINT "Gioca un numero intero compreso tra 1 e 3. Hai capito?" : GOTO 140
160 M=M+N
170 CLS
180 PRINT "Il totale e'";M
190 N=4-N
200 M=M+N
210 IF M=15 THEN PRINT"Ho scelto";N;"e HO VINTO!" : END
220 PRINT "Ho scelto";N;". Il totale e'";M;". Adesso    scegli tu"
230 GOTO 140
240 '-------------------------------------------------------------------------------------------------------
250 PRINT "Scegli il numero"
260 INPUT N
270 CLS
280 IF FIX(N))(N OR N(1 OR N)3 THEN PRINT "Gioca un numero intero compreso tra 1 e 3. Hai capito?" : GOTO 260
290 M=M+N
300 PRINT "Il totale e'";M
310 IF M=15 THEN  PRINT "e HO PERSO!" : END
320 IF N=1 THEN 340 ELSE 390
330 '-------------------------------------------------------------------------------------------------------
340 N=2
350 M=M+2
360 IF M=15 THEN PRINT "Ho scelto 2. Il totale e' 15 e HO VINTO!" : END
370 PRINT "Ho scelto 2. Il totale e'";M;". Adesso    scegli tu"
380 GOTO 140
390 IF N=2 THEN 410 ELSE 470
400 '-------------------------------------------------------------------------------------------------------
410 N=1
420 M=M+1
430 IF M=15 THEN PRINT "Ho scelto 1. Il totale e' 15 e HO VINTO!" : END
440 PRINT "Ho scelto 1. Il totale e'";M;". Adesso    scegli tu"
450 GOTO 140
460 '-------------------------------------------------------------------------------------------------------
470 N=1
480 M=M+1
490 PRINT "Ho scelto 1. Il totale e'";M;". Adesso    scegli tu"
500 GOTO 260
```


Al programma "Fatturazione e Dichiarazione IVA" per M 10 (numero 1 di *Libreria di Software*) vanno apportate le seguenti modifiche.
È necessario trascriverle a programma caricato, prima di dare il comando RUN.
Si inseriranno automaticamente al posto giusto.

```
550  IF B$="S" OR B$="s" THEN CLOSE£2 : GOTO 120

3510 PF£ = PF£+(F1£+F2£+F3£+F4£+F5£) : GOTO 3440

3082 IF S%=2 THEN IF DW%<=DY% AND (DN%>3 AND DN%<=DM%)THEN3410ELSE3050

3083 IF S%=3 THEN IF DW%<=DY% AND (DN%>6 AND DN%<=DM%)THEN3410ELSE3050

3491 IF S%=2 THEN IF DX%<=DY% AND (DR%>3 AND DR%<=DM%)THEN3510ELSE3440

3492 IF S%=3 THEN IF DX%<=DY% AND (DR%>6 AND DR%<=DM%)THEN3510ELSE3440
```

Per Commodore 64 e Spectrum il caricamento del programma deve avvenire col comando LOAD "nome programma".

Per eventuali informazioni riguardanti le cassette e la Libreria di Software scrivere a:

— Claudio Parmelli V.tto cieco B.go Tascherio, 9 - 37129 VERONA

# Banca Aperta

## LE NUOVE RISPOSTE DEL BANCO DI ROMA.

*Vorrei avere un rapporto più diretto con la mia banca...*

Anche le strutture bancarie si evolvono. Il Banco di Roma, primo in Italia, sta introducendo la struttura a "banca aperta", già attuata da molte sue filiali italiane. "Banca aperta": non il solito bancone, le lunghe file, ma un nuovo modo di essere banca, un rapporto più personalizzato, un clima più agevole, più professionale e una maggiore rapidità in ogni operazione. Un ulteriore passo avanti verso la completa consulenza finanziaria che il Banco di Roma intende mettere a disposizione dei propri clienti. Tra i numerosi servizi offerti ricordiamo: Prestito Personale, Prestito Casa, gestione dei patrimoni, Leasing, assistenza all'import-export, attraverso ben 60 sedi estere in 30 Paesi dei 5 continenti. Tutto questo perché il Gruppo Banco di Roma è in grado di gestire ogni servizio specifico con grande professionalità, fornendo anche informazioni dirette a domicilio attraverso i sistemi Videotel e Voxintesi.


**BANCO DI ROMA**

CONOSCIAMOCI MEGLIO.