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

26 CORSO PRATICO COL COMPUTER

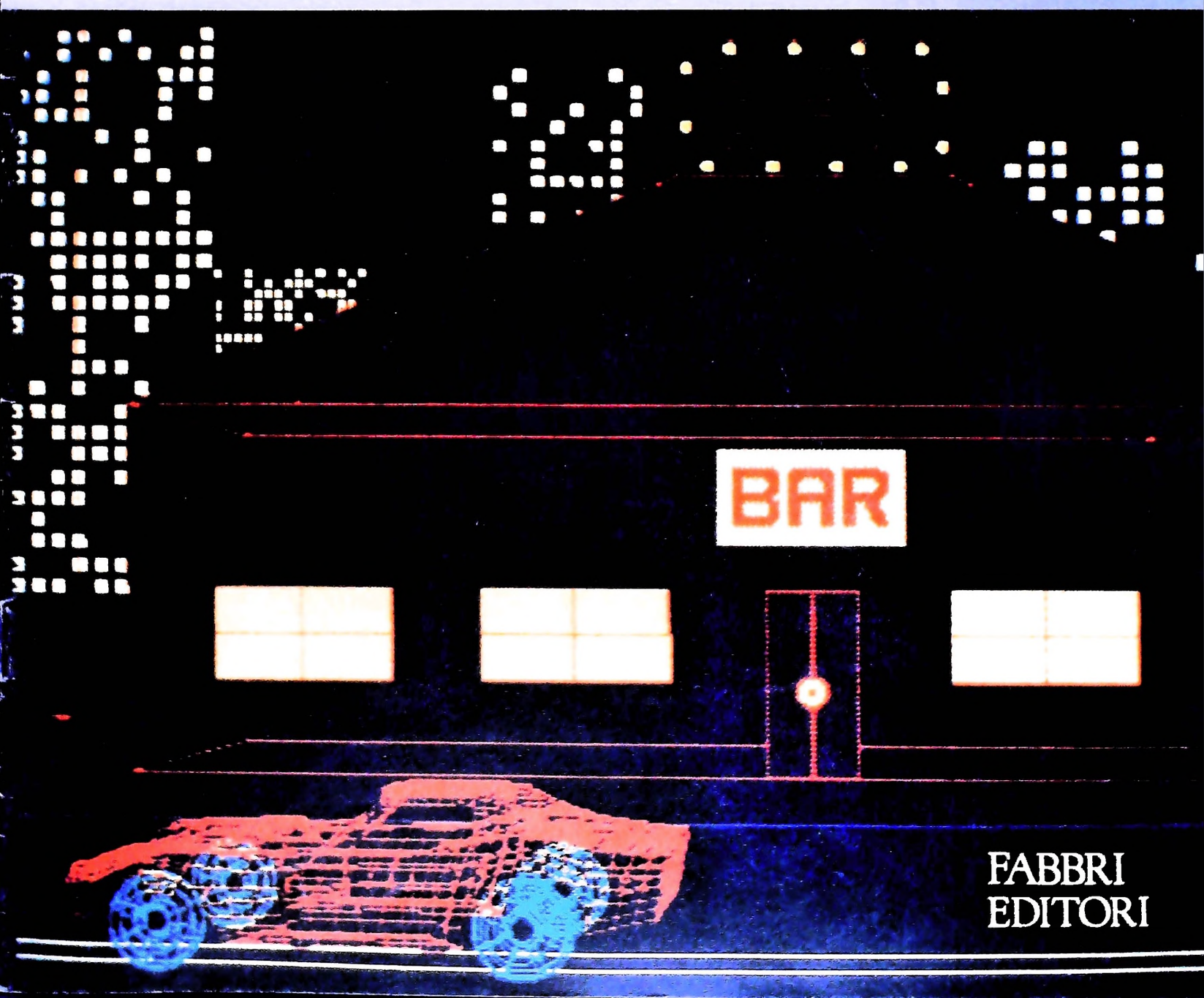
421784

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

F4 F5 F6 F7 F8

BATTERY LOW

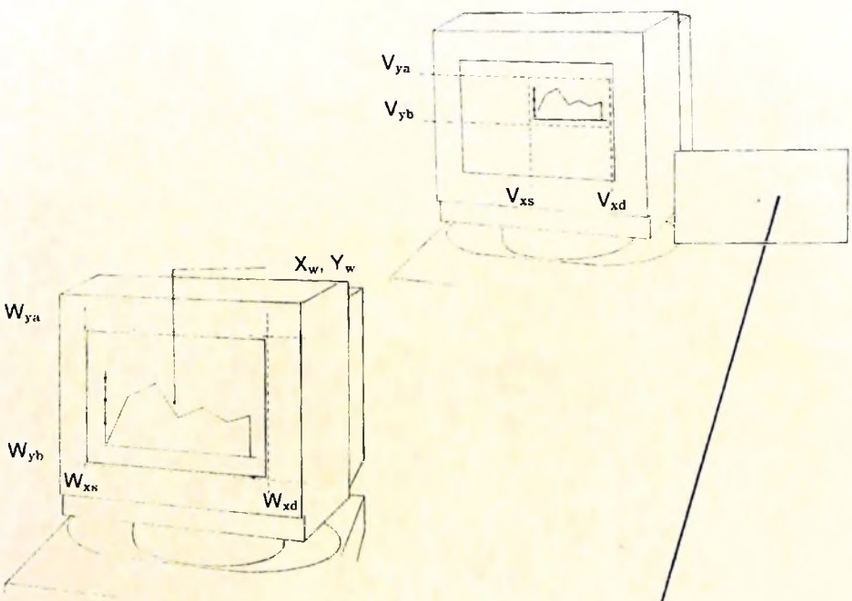
diretto da GIANNI DEGLI ANTONI



FABBRI EDITORI

ERRATA CORRIGE

Nell'articolo "I pacchetti grafici" del fascicolo n. 13, alle pagg. 210-211, i bordi della window e quelli della viewport vanno così corretti:



per cui le formule diventano:

$$X_s = \frac{V_{xd} - V_{xs}}{W_{xd} - W_{xs}} (X_w - W_{xs}) + V_{xs}$$

$$Y_s = \frac{V_{ya} - V_{yb}}{W_{ya} - W_{yb}} (Y_w - W_{yb}) + V_{yb}$$

e il termine frazionario:

$$(V_{xd} - V_{xs}) / (W_{xd} - W_{xs})$$

Corso Pratico col Computer - Copyright © sul fascicolo 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Copyright © sull'opera 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Prima Edizione 1984 - Direttore responsabile GIOVANNI GIOVANNINI - Registrazione presso il Tribunale di Milano n. 135 del 10 marzo 1984 - Iscrizione al Registro Nazionale della Stampa n. 00262, vol. 3, Foglio 489 del 20.9.1982 - Stampato presso lo Stabilimento Grafico del Gruppo Editoriale Fabbri S.p.A., Milano - Diffusione Gruppo Editoriale Fabbri S.p.A. via Mecenate, 91 - tel. 50951 - Milano - Distribuzione per l'Italia: A. & G. Marco s.a.s., via Fortezza 27 - tel. 2526 - Milano - Pubblicazione periodica settimanale - Anno I - n. 26 - esce il giovedì - Spedizione in abb. postale - Gruppo II/70. L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

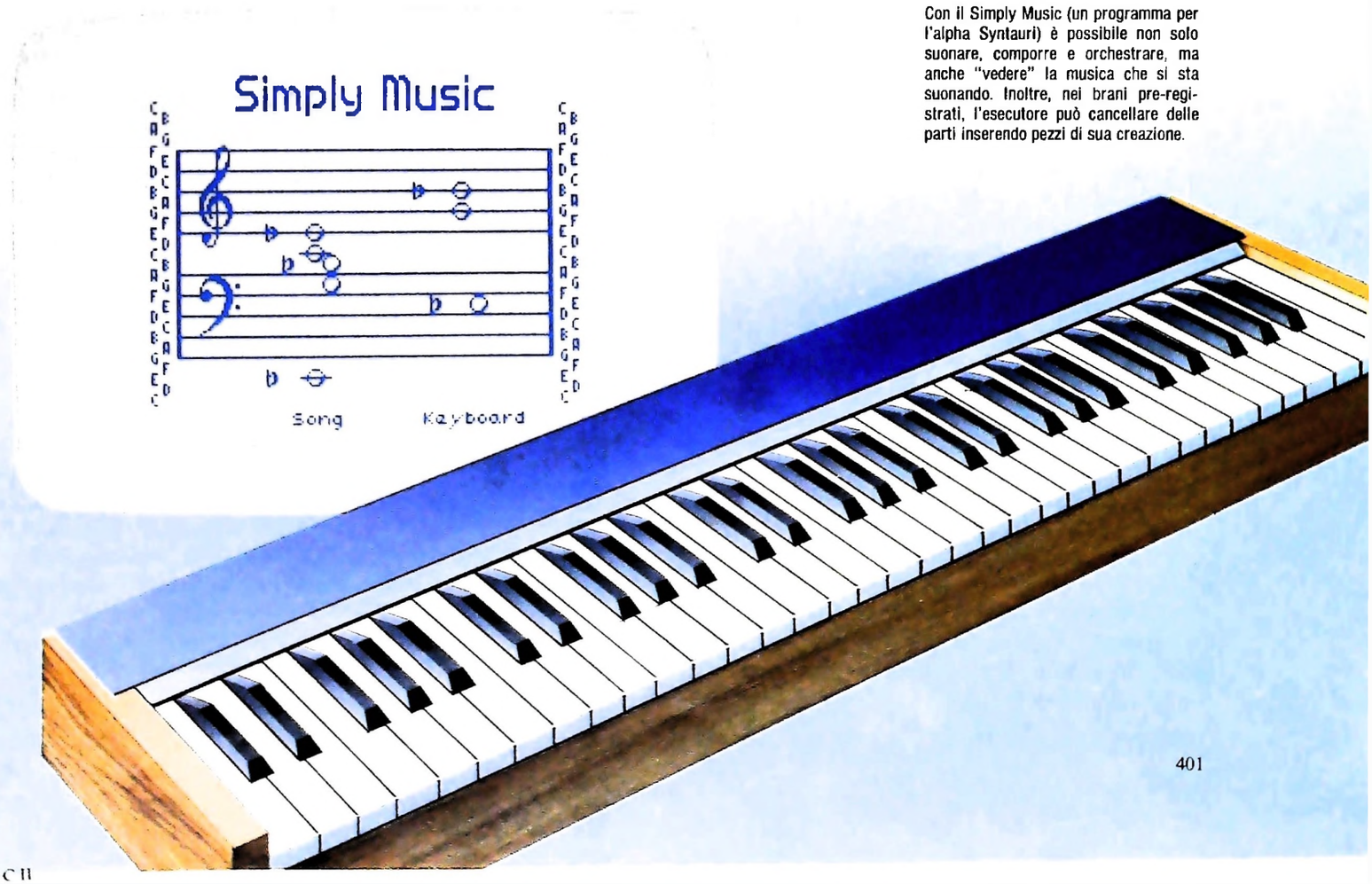
IL SISTEMA PER L'ELABORAZIONE MUSICALE (II)

Esaminiamo i criteri per l'organizzazione delle informazioni e le limitazioni imposte dal personal computer.

A conclusione di questa rubrica completiamo il discorso sul sistema per l'elaborazione musicale (SEM), considerando, gli aspetti legati alla performance musicale, i diversi criteri per l'organizzazione delle informazioni del SEM e, infine, le limitazioni che vengono imposte dalla scelta del personal/portable computer come strumento per l'elaborazione musicale.

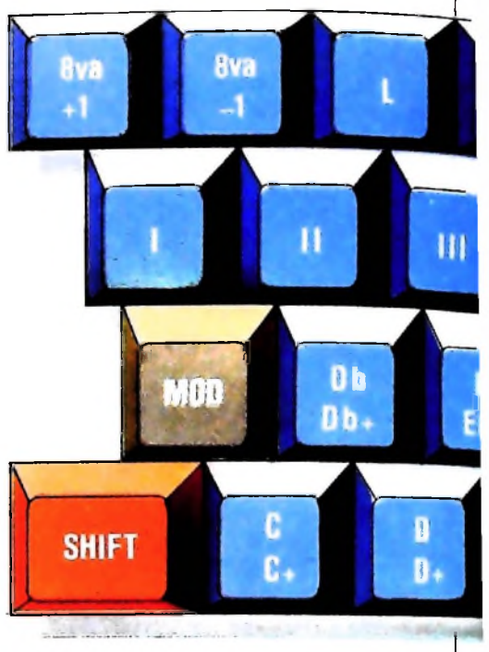
Performance musicale e configurazione della tastiera

Parlando di performance intendiamo tutte quelle attività musicali connesse con la produzione e l'analisi dei suoni in tempo reale: esecuzione, composizione improvvisata ecc.
Abbiamo visto nel precedente numero come sia questa l'attività più esigente in termini di risorse di calcolo.
Un aspetto rilevante, che condiziona in modo determinante le modalità e la qualità della performance, riguarda i dispositivi per comunicare al SEM i comandi musicali.
Il limite di molti strumenti musicali elettronici molto sofisticati e potenti risiede spesso nella mancanza di appropriati dispositivi (e linguaggi) per comunicare *in breve* i comandi musicali.
Il gesto di scegliere un tasto del pianoforte e premerlo con una certa intensità e per un certo tempo determina la produzione di una *nota* con tutti i suoi parametri: l'altezza (posizione del tasto), l'intensità (pressione esercitata sul tasto), la durata (intervallo di tempo tra l'inizio e la fine della pressione del tasto) e il timbro (scelta del particolare strumento, il pianoforte in questo esempio).

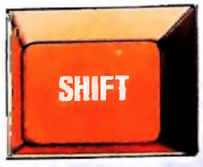


Con il Simply Music (un programma per l'alpha Syntauri) è possibile non solo suonare, comporre e orchestrare, ma anche "vedere" la musica che si sta suonando. Inoltre, nei brani pre-registrati, l'esecutore può cancellare delle parti inserendo pezzi di sua creazione.

Esempio di configurazione musicale della tastiera ASCII.



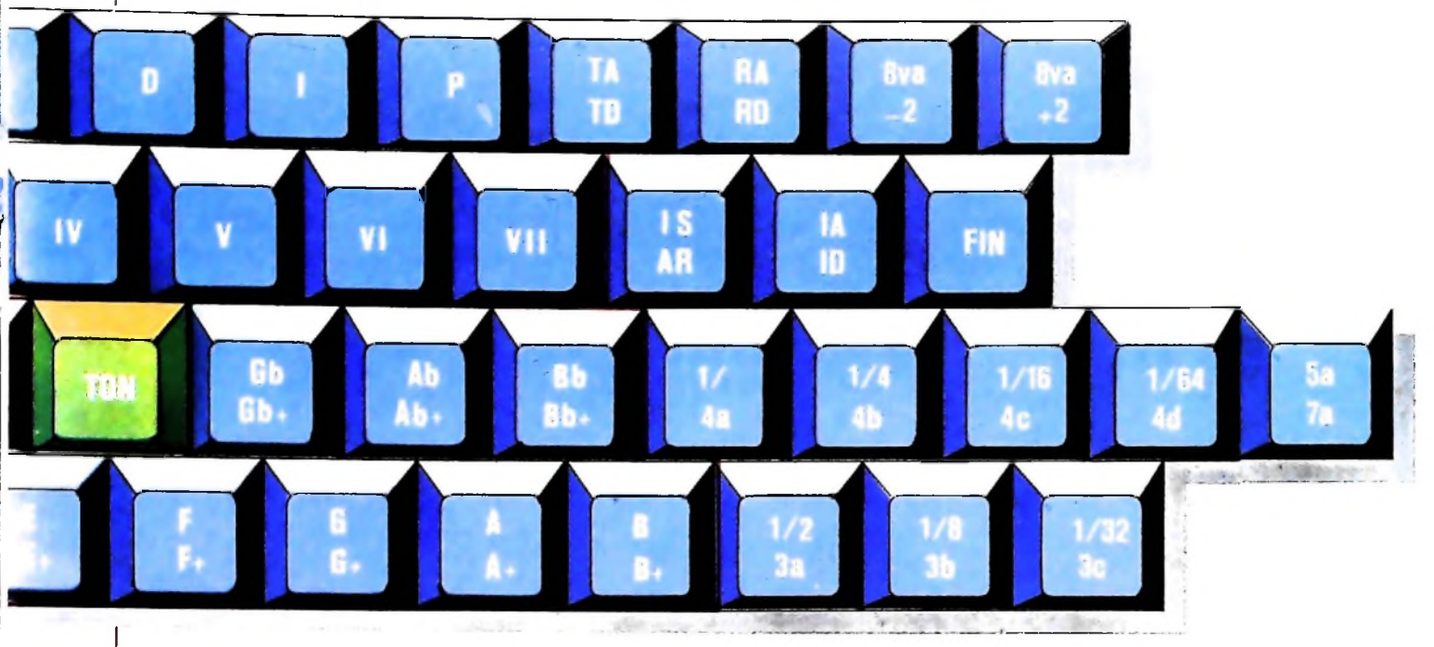
Questo tasto premuto permette di selezionare i dodici semitoni dell'ottava, rappresentata dai dodici tasti. Lasciandolo libero, i tasti selezionano i dodici semitoni trasposti di un quarto verso il tono acuto.



Il TON consente di scegliere tra l'uso di una scala temperata e l'uso di una scala naturale. Questo tasto ha quindi la possibilità di funzionare da switch, cioè da "ago indicatore".

Con questo tasto si giunge all'ultimo passaggio, cioè dalla performance all'editing: la tastiera viene usata per modificare o eseguire o stampare un testo musicale prima registrato.

In breve, mediante un pianoforte possiamo eseguire sequenze di note e sovrapporre tante sequenze quante siamo capaci di controllare con le dieci dita delle mani; la performance è inoltre semplificata dal fatto che il timbro è *predefinito* nella struttura fisica e meccanica dello strumento stesso.
Invece, di fronte a uno strumento elettronico (che permette la costruzione del timbro) ci troviamo a dover definire le caratteristiche timbriche di ogni suono, anche se (con una procedura semplificata, ma limitante) possiamo definire all'inizio di una performance i timbri che serviranno.
D'altra parte, diventa possibile comunicare con l'elaboratore anche a livelli più astratti di quello del pianoforte; possiamo comandare l'esecuzione di funzioni musicali, trasformare parte della performance già eseguita o di materiale sonoro preregistrato, elaborare un suono "ascoltato" dal SEM ecc.
Possiamo immaginare una sessione al SEM come costituita dai seguenti passi:
a) definiamo i timbri (cioè la nostra orchestra sintetica);
b) definiamo delle sequenze di comandi o di eventi musicali che useremo durante la performance (il nostro repertorio di strutture musicali di base);
c) eseguiamo o improvvisiamo un brano musicale usando una tastiera (che potrà essere sia musicale tipo pianoforte, che una comune tastiera ASCII di elaboratore) per la definizione in tempo reale dei parametri di altezza, di intensità e di durata e per il richiamo dei timbri e delle strutture musicali precedentemente definite.
In questo modo abbiamo un'accresciuta potenza di comunicazione con l'elaboratore rispetto alle possibilità offerte dagli strumenti tradizionali poiché possiamo richiamare ed eseguire strutture musicali complesse quanto vogliamo con un solo gesto! E non abbiamo l'onere e la difficoltà (ben nota a chi ha avuto esperienze con i sintetizzatori analogici) di dover specificare le informazioni musicali solo al livello operativo dello strumento.
Possiamo ipotizzare, ad esempio, una tastiera ASCII (figura in alto in queste pagine). Il riquadro rosso in basso comprende dodici tasti corrispondenti alle altezze di un'ottava temperata in dodici semitoni; in maiuscolo (tasto SHIFT premuto) sono i dodici semitoni dell'ottava, in minuscolo sono i dodici semitoni trasposti di un quarto di tono verso l'acuto. Così abbiamo una tastiera per quarti di tono.
Il riquadro rosso in alto è costituito da sette tasti corrispondenti ai sette gradi della scala diatonica naturale; mediante il tasto TON scegliamo se operare con la scala temperata o con la scala naturale, cioè il tasto TON funziona da switch; in questo secondo caso, dovremo scegliere la tonalità premendo uno dei dodici tasti del riquadro rosso inferiore e quindi potremo operare sui tasti dei gradi della tonalità prescelta.
Sulla destra del riquadro inferiore abbiamo la sezione ritmica della tastiera: i tasti 1, 1/2, ... , 1/64 ci permettono di selezionare il valore di durata della nota che ci accingiamo a suonare; la selezione della durata non può essere implicita nel gesto (cioè dipendente dal tempo con cui si esercita la pressione sul tasto) poiché non è previsto un tale dispositivo nella tastiera. I tasti 4a, 4b, 4c, 4d servono invece a selezionare un ritmo (tra quattro) in quattro quarti che si ripete fino a che non interveniamo sulla sezione ritmica della tastiera selezionando un altro ritmo o un particolare valore di durata. I tasti 3a, 3b, 3c selezionano ritmi in 3/4; i tasti 5a e 7a sele-



zionano ritmi in 5/4 e 7/4 rispettivamente.
La prima linea di tasti comprende i quattro salti d'ottava previsti dalla tastiera: 8va+1 (un'ottava sopra), 8va-1 (un'ottava sotto), 8va-2 (due ottave sotto), 8va+2 (due ottave sopra).
I tasti TA/TD (trasposizione altezza/durate), RA/RD (retrogradazione altezze/durate), IA/ID (iterazione altezza/durate), IS/AR (inversione speculare altezze/rallentando-accelerando) sono funzioni che possiamo applicare alle sequenze musicali generate durante la nostra performance: così possiamo ripetere un ritornello per un certo numero di volte, o rovesciarlo rispetto al tempo o rispetto agli intervalli, o ancora ripetere una certa struttura ritmica (uguale o trasformata) ecc. Questi comandi innalzano il livello di comunicazione rendendo la performance estremamente più potente.
Il tasto MOD è uno switch che ci serve per passare dalla modalità di performance alla modalità di editing in cui usiamo la tastiera per leggere, modificare o eseguire o stampare un testo musicale preregistrato (eventualmente proprio la performance che abbiamo appena eseguito e registrato sulla memoria dell'elaboratore). I tasti L (locate), C (change), D (delete), I (insert), P (print) servono appunto ad eseguire le funzioni di editing: trovare una certa sequenza, cambiare una sequenza, cancellare una sequenza, inserire in una sequenza, stampare ed eseguire la codifica di una sequenza. Per ritornare in modalità di performance ci basterà premere nuovamente il tasto MOD.
Infine, il tasto FIN conclude la sessione musicale.
Oltre alla tastiera, abbiamo altri dispositivi che arricchiscono le possibilità di comunicazione durante la performance: pedali, cursori, joystick, light-pen ecc.
Quello che più è interessante nel SEM è il poter associare, mediante un opportuno programma, diverse funzioni di volta in volta ad ogni dispositivo; ad esempio, il pedale può essere usato per il controllo dell'intensità o per passare gradualmente da un timbro ad un altro o per controllare il temperamento dell'ottava o per altre funzioni ancora.
La riconfigurabilità dei dispositivi vale naturalmente anche per la tastiera e questo è veramente molto importante. Una sola tastiera ha in questo modo illimitate funzionalità che le vengono associate sulla base della preventiva definizione di differenti configurazioni in cui ogni tasto ha una funzione; ogni configurazione corrisponde ad un ambiente di comunicazione: cambiando l'ambiente cambiano il linguaggio e le possibilità di espressione sonora.
Un aspetto pratico per rendere meglio fruibile la riconfigurabilità è la possibilità di preparare delle mascherine (o delle etichette adesive) da applicare sulla tastiera in corrispondenza con le diverse configurazioni per rendere immediato il riconoscimento del significato associato a ogni tasto nelle diverse configurazioni della tastiera.

Organizzazione dell'informazione musicale

La quantità di memoria disponibile e i tempi di accesso alle informazioni memorizzate condizionano le architetture e le funzionalità del SEM.
Dovremo per prima cosa distinguere tra i SEM che dispongono di memoria permanente e quelli che perdono tutte le

informazioni quando vengono spenti: i secondi potranno essere usati solo per performance, i primi anche per elaborazioni compiute in varie fasi. Dobbiamo anche distinguere tra i SEM che dispongono di memorie di massa (ad esempio, il registratore a cassette per l'M10) e i SEM che hanno solo memoria centrale, poiché solo con un'opportuna memoria di massa possiamo mantenere l'informazione relativa ai testi musicali e (a maggior ragione) ai suoni; solo un'ampia memoria di massa ci consente di gestire un archivio musicale.
In dipendenza dell'architettura hardware e delle funzionalità previste da un SEM potremo avere differenti soluzioni per l'organizzazione dell'archivio musicale; in sintesi possiamo distinguere tre modelli organizzativi:
a) SEM in cui l'informazione musicale è memorizzata a un solo livello di rappresentazione;
b) SEM in cui l'informazione musicale è memorizzata in modo esplicito ai diversi livelli di rappresentazione, in regime di copresenza;
c) SEM in cui l'informazione musicale è memorizzata in una forma mista tra i livelli di rappresentazione, dotati di strumenti automatici per convertire la rappresentazione a un livello in quella al livello superiore (*analisi*) o al livello inferiore (*sintesi*).
Il primo tipo rende molto difficile la comunicazione musicista-SEM (come accadeva con la strumentazione analogica). Il secondo tipo è molto oneroso perché richiede memoria e numerosi calcoli per il mantenimento in parallelo dei diversi livelli di rappresentazione. Il terzo tipo è senz'altro il più ragionevole ed efficace poiché non pone limitazioni a priori e non richiede risorse in ridondanza.

Esecuzione sull'alpha Syntauri, un sistema completo formato di tastiera e software apposito, che permette di elaborare le forme d'onda, registrare su 16 piste e utilizzare nello stesso tempo voci diverse, visualizzando i brani sullo schermo. Il musicista alla tastiera è il jazzista americano Chick Corea.

Limitazioni dell'architettura funzionale del SEM sui personal computer

Implementare un SEM su un personal computer richiede la soluzione di alcuni problemi e la limitazione, in termini funzionali, del SEM generale, cioè il SEM capace di qualunque elaborazione del testo musicale e del suono.
Consideriamo tre tra i più diffusi personal computer: l'Olivetti M10, il Sinclair Spectrum e il Commodore 64.
L'M10, mediante l'istruzione SOUND, ci permette di descrivere testi musicali rispetto ai parametri di altezza e di durata; sintetizza il suono mediante un oscillatore, quindi è monofonico.
Lo Spectrum usa l'istruzione BEEP che, concettualmente, ci offre le medesime possibilità dell'M10: descrizione dell'altezza, della durata e sintesi mediante un oscillatore.
Il Commodore 64, oltre al controllo delle altezze e delle durate, ci consente di controllare l'intensità dei suoni anche mediante un inviluppo di tipo ADSR (attack-decay-sustain-release); inoltre, disponendo di tre oscillatori, ha una polifonia a tre voci e il timbro è parzialmente controllabile (mediante selezione tra un insieme di forme d'onda prefissate).
In breve, le attività di analisi, elaborazione e sintesi del testo musicale sono implementabili su tutti i personal, con gli ovvi limiti di dimensione dei testi in dipendenza dalla quantità di memoria centrale e di massa disponibile.
L'analisi del suono non è possibile, non disponendo i personal computer di strumenti per l'acquisizione di segnali audio; l'elaborazione di segnali è teoricamente possibile, ma non potendo né analizzare né sintetizzare suoni numericamente non ha molto interesse; inoltre richiede molto tempo di calcolo. La sintesi digitale del suono non è possibile (non disponendo i personal computer di convertitori D/A), ma è possibile la sintesi sonora mediante gli oscillatori; si tratta quindi di sistemi ibridi, privi del sottosistema di acquisizione dei segnali audio.

MINI WORD PROCESSOR

Un programma che permette di fissare i margini della pagina, il numero di righe, i capoversi.

Il programma TEXT contenuto nell'M10 permette di definire solo la dimensione della riga.
L'uso del Mini Word Processor permette di implementare alcune delle più comuni funzioni dell'elaborazione testi:
1. Definizione del margine sinistro
2. Definizione del margine destro
3. Definizione del margine alto
4. Lunghezza della pagina stampata
5. Numerazione delle pagine
6. Intestazione delle pagine (seguenti la prima)
7. Opzione di interlinea per stampanti sprovviste di LINE FEED automatico
8. Centratura delle righe
9. Opzione di nuova pagina
10. Giustificazione a destra della riga desiderata.

Le prime sette opzioni possono essere utilizzate con qualsiasi file ASCII (memorizzato con estensione .DO), mentre le 8, 9, 10 richiedono l'aggiunta di particolari caratteri di controllo nel testo.
Il simbolo grafico per specificare che il carattere che lo segue si riferisce a un controllo è il "^", posto sotto il numero 8.
Caricato il programma e digitato il RUN, si richiede di definire i parametri di stampa, originariamente così posizionati:

```
Margine sinistro ? 0          Margine destro ? 60
Margine alto ? 4
Righe per pagina ? 54
Numerazione da pagina ? 0
Intestazione ?
Vuoi interlinea forzata ? N
```

Nel caso si voglia mantenere la configurazione di base, sarà sufficiente premere il tasto ENTER a ogni richiesta; in caso contrario digitare il numero o la frase prescelta.
Se, per esempio, si volesse un documento centrato su di un foglio a ottanta colonne, sarà necessario posizionare il margine sinistro a 10 e quello destro a 70, ottenendo di conseguenza una riga di 60 caratteri.
La lunghezza della pagina è in origine dimensionata al massimo di righe consentito. Oltre le 54 righe si potrebbe incorrere in errori di posizionamento del nuovo foglio di carta; è invece possibile abbassare a piacere tale limite. È utile ricordare che per un normale formato di carta a 66 righe quale quello in uso per le stampanti a modulo continuo, 54 righe per pagina e un margine alto di 4 è il dimensionamento consigliato per un miglior inquadramento di un documento multipagine.
Se la numerazione di pagina è lasciata a zero, il numero non verrà né incrementato né stampato sulle pagine, in caso contrario invece verrà incrementato e stampato dalla seconda pagina in poi. Se esiste intestazione, la stessa verrà stampata a partire dal secondo foglio in poi e giustificata sul margine sinistro se e solo nel caso in cui l'opzione di stampa del numero di pagine è abilitata.

Creazione del documento

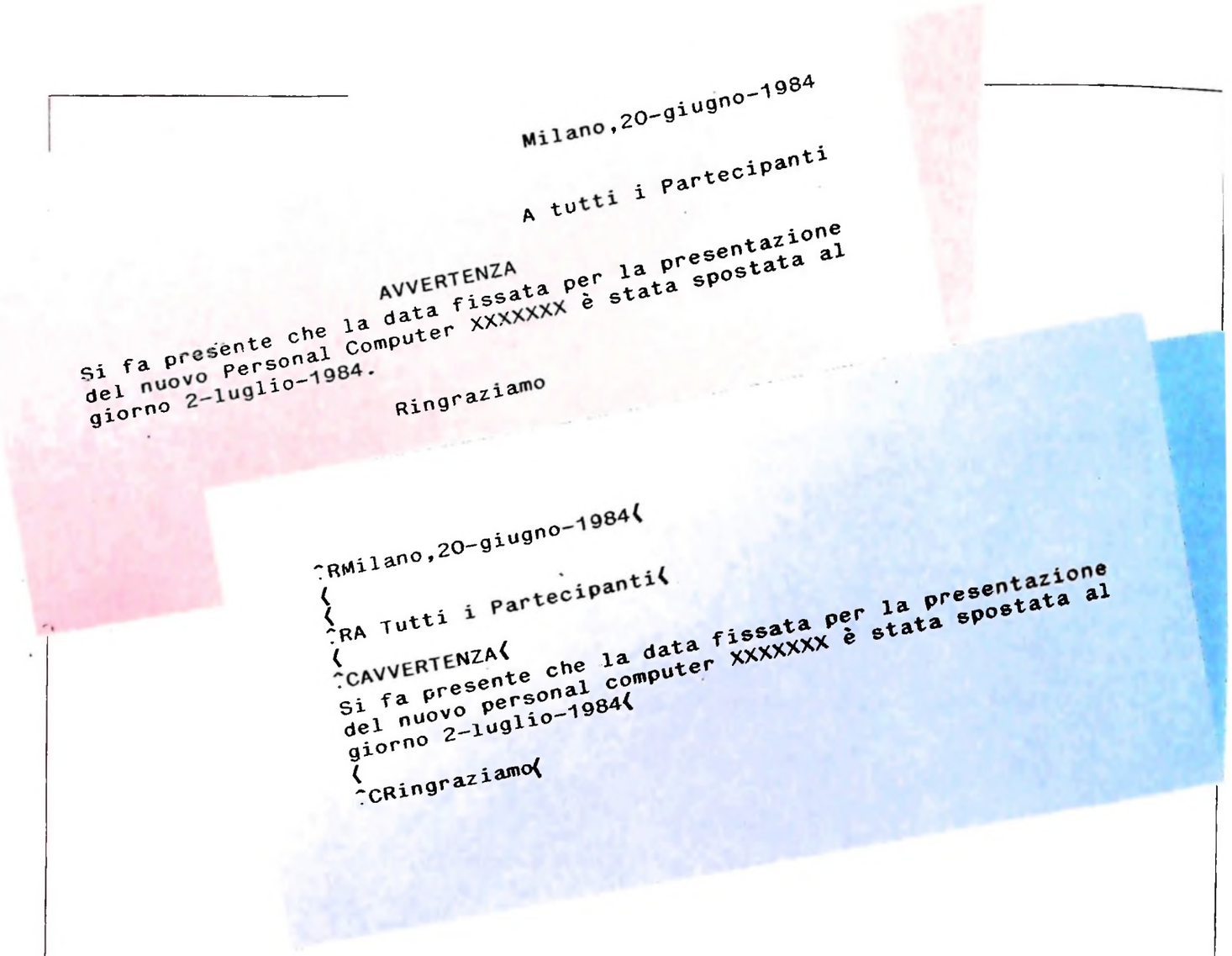
Per la creazione del documento che dovrà essere elaborato dal Mini Word Processor è sufficiente utilizzare il programma TEXT ricordando che se si desidera aumentare l'interlinea o se si vuole spaziare i paragrafi tra loro è necessario premere più volte il tasto ENTER (una pressione = una interlinea). Questo perché né il TEXT né il Mini Word Processor possiedono, per ora, l'opzione di scelta della spaziatura tra le righe.
Per far rientrare un paragrafo è necessario premere la barra spaziatrice tante volte quanti sono gli spazi che vogliamo lasciare vuoti. È consigliabile non utilizzare il tasto TAB dato che lo stesso ha la funzione di spostare il cursore di otto spazi sul display, ma non su tutte le stampanti.

Caratteri di controllo

Sono tre i codici utilizzabili in qualsiasi testo per permettere differenti caratteristiche di stampa:

^F per passare a nuova pagina
^C per centrare la riga
^R per giustificare a destra una riga.

Normalmente, il programma salta alla nuova pagina non appena raggiunge il numero di righe specificato all'atto del dimensionamento, ma se si volesse anticipare tale funzione, è sufficiente far precedere il testo che si vuole spostare dal codice ^F.
Per eseguire la funzione di centratura è necessario far precedere il testo dal codice ^C (il testo dovrà avere lunghezza inferiore alla dimensione della riga). Per abilitare la funzione



di giustificazione basta far precedere il testo dal codice ^R. Se, per esempio, si volesse ottenere una lettera come quella riportata in alto in questa figura, bisognerebbe comporre il documento corrispondente nella forma visibile nella stessa illustrazione, in basso, dove con il carattere ⟨ si è simboleggiata la pressione del tasto ENTER.

Il programma

Mini Word Processor è sostanzialmente basato sulla manipolazione di una stringa. Per meglio comprendere l'analisi e il funzionamento di questo programma simuliamo l'elaborazione del seguente testo:

M10 IL⟨

^CPORTATILE OLIVETTI⟨

^RPIÙ PICCOLO.⟨

dimensionando la stampa secondo i parametri:

MARGINE SINISTRO 10

MARGINE DESTRO 30

MARGINE ALTO 4

RIGHE PER PAGINA 54

NUMERAZIONE DA PAGINA 0

INTESTAZIONE

VUOI INTERLINEA FORZATA? N

LINEA 10 - Pulisce il video, dimensiona a 2 il numero massimo di file che verranno trattati nel corso dell'elaborazione (il primo di INPUT della memoria ed il secondo di OUTPUT sulla periferica), stabilisce che le variabili inizianti con le lettere I,A,B,C,D,E siano considerate stringhe mentre quelle inizianti per P,L,R,X,Y,Z siano considerate interi.

LINEA 20 - Stabilisce le due variabili utilizzate per visualizzare le scritte in negativo sul display LCD.

LINEA 30 - Richiama la subroutine 2000 che svolge la funzione di temporizzazione.

LINEA 40 - Normalizza le variabili utilizzate per la manipolazione delle stringhe (RP,LP), il puntatore ZP ed il contatore di righe LC.

LINEA 50 - Salta alla 180.

LINEE 180-220 - Richiedono la definizione dei parametri:
XL = margine sinistro XR = margine destro
FL = margine alto PL = righe per pagina

LINEA 230 - Calcola il valore di LL che, derivante dalla differenza tra XR e XL, altro non è che il numero di caratteri per riga + 1 (+ 1 perché leggendo il file carattere per carattere viene letto anche il carattere ENTER che non viene visualizzato in stampa ma solo eseguito). Crea la stringa di base formata da un numero di spazi uguale a LL; nel caso in esame, dal momento che XR=30 e XL=10 si ha di conseguenza che LL=21, quindi 10= "....." (21 spazi).

LINEE 240-270 - Richiedono le opzioni supplementari quali la numerazione di pagina XN, l'intestazione CT e la forzatura di interlinea LF$ nel caso la stampante utilizzata non contempli il LINE FEED automatico dopo il Carriage Return (come nel caso della PR320 OLIVETTI).

LINEE 300-340 - Visualizza i file presenti nella memoria dell'M10, richiede il nome del documento da trattare ed apre i file corrispondenti per poi portare l'esecuzione alla riga 90.

LINEA 90 - Richiama la subroutine di marginatura alta, facendo "stampare" tante righe vuote quante il valore della variabile FL.

LINEA 100 - Controllo sulla fine del file.

LINEE 110-120 - Legge il primo carattere del documento (la funzione INPUT$(1,1), esegue la lettura sul file 1 di un carattere alla volta), in questo caso M, associandolo alla variabile I, se lo stesso risulta diverso da caratteri speciali quali il CR,LF, " ", prosegue.

LINEA 150 - Pone il puntatore ZP=0 e sostituisce all'RP--esimo carattere della stringa 10 il contenuto della variabile I: essendo RP=1 ne consegue che la stringa 10 sarà uguale a "M.....".

LINEA 160 - Incrementa la variabile RP (che ora sarà 2) rappresentante il numero di caratteri letti + 1 e controlla che tale valore non sia superiore alla dimensione LL data della riga. In tal caso passa alla linea 170 che rimanda l'elaborazione alla riga 100.

L'esecuzione continua in maniera analoga fino a che la stringa 10 non sia uguale a "M10 IL.....". A questo punto per effetto della linea 110, il carattere I letto risulta uguale a CHR$(13) cioè un ritorno a capo (ENTER), questo causa il salto alla riga 2400.

LINEE 2400-2410 - Pone LP=RP essendo RP=7 (per quanto visto prima), ne risulta LP=7 e ZP=1 e controlla che LC sia inferiore al numero di righe per pagina (PL) fissato.

LINEA 2420 - Sposta la testina di stampa di XL spazi vuoti (il margine sinistro) quindi scrive i primi 6 caratteri (LP-1) della stringa 10 e quindi M10 IL. A questo punto se è presente l'opzione di interlinea forzata invia la stringa LF$ che è il codice di LINE FEED, se invece tale funzione è automatica l'operazione non viene eseguita, in ambedue i casi LC viene incrementato di 1.

LINEA 2430 - Pone nella variabile W$ l'LP-esimo carattere della stringa 10, essendo LP=7 ne risulta W$= " ", quindi riporta la variabile 10="....."
L'utilità della routine 2430-2500 è più evidente nel caso di stringa più lunga della dimensione riga fissata. Questa ipotesi verrà discussa più avanti.

LINEA 2460 - Normalizza RP e LP e pone l'elaborazione alla riga 100 la quale effettua il consueto controllo sulla fine del file.

LINEA 110 - Legge il carattere che esegue e lo associa alla variabile I; essendo I="^" l'elaborazione viene indirizzata alla linea 350

LINEA 350 - Pone ZP=0 controlla il valore di RP per assicurarsi che il simbolo "^2 sia effettivamente il primo carattere analizzato e quindi carattere di controllo.

LINEA 360 - Associa alla variabile I il carattere seguente, per cui I="C". Ne consegue la lettura (da parte della linea 370) di tutta la riga del file e quindi per effetto dell'associazione, I ="PORTATILE OLIVETTI". Sposta in seguito la testina di stampa di tanti spazi quanti risultanti dall'operazione ((LL-LEN(I))/2+XL); nel caso in esame essendo LL=21,LEN(I)= 18,XL=10 risulta 11, per cui in dodicesima colonna verrà stampata la stringa "PORTATILE OLIVETTI" che si troverà centrata rispetto ai margini prefissati. Susseguentemente verrà incrementato LC e l'elaborazione si sposterà ancora alla riga 100 dove effettuerà il solito controllo di fine file.

LINEA 110 - Legge il carattere seguente che essendo "^" causerà l'intervento della riga 350 che controllerà il valore di RP.

LINEA 380 - Essendo I="R", verrà assegnata alla variabile I tutta la riga del file e quindi I="PIÙ PICCOLO", sposterà, in seguito, la testina di stampa di tanti spazi quanti risultanti dall'operazione XR-LEN(I) e quindi la variabile I, la stringa

"PIÙ PICCOLO" si troverà a essere giustificato a destra come richiesto.
Riportata l'elaborazione alla riga 100 e constatata la fine del file, la stessa si sposterà alla riga 2400 dove dopo elaborazione ininfluente alla linea 2460 chiuderà il file e il programma.
Come accennato in precedenza, analizziamo in dettaglio la routine 2410-2460 nel caso in cui sia RP› › LL.
Poniamo quindi il caso di dover trattare la frase "OLIVETTI M10" avendo dimensionato quindi XL=10 e XR=20 (LL=11).
Accadrà che fino alla costituzione della stringa 10="OLIVETTI M1" tutto procederà come prima fermo restando che la LINEA 150 porrà LP=RP non appena I= " " e questo accadrà quando RP=9. Ad un certo punto la condizione di cui alla linea 160 risulterà vera per cui l'elaborazione verrà indirizzata alla linea 2410 dove assicuratici di non aver oltrepassato il numero di righe prefissate si continuerà con la seguente riga.

LINEA 2420 - Dopo aver spostato la testina di stampa di XL spazi, stamperà la stringa composta dai primi LP-1 carattere di 10 e quindi essendo LP=9 verrà stampata la sequenza "OLIVETTI" seguita dal LINE FEED automatico. Viene quindi incrementato LC.

LINEA 2430 - Viene creata la stringa W$ che per effetto dell'espressione che la forma assumerà il valore "M10", mentre la stringa 10 verrà ricomposta da LL spazi.

LINEA 2440 - L'effetto della riga farà in modo che la stringa W$ diventi "M10" (viene annullato lo spazio in testa).

LINEA 2450 - Sostituisce la stringa W$ ai primi tre (LEN(W$)) caratteri di 10 e posizionerà il puntatore RP a LEN(W$)+1 cioè 4, in modo che il seguente carattere letto verrà assunto in coda alla stringa "M10" sulla nuova riga di stampa.

La routine 2500 permette la stampa dell'intestazione e del numero di pagina.

```
5 '************************************        ************************************
7 '*                                                                              *
8 '*                          MINI WORD PROCESSOR                                 *
9 '************************************        ************************************

10 CLS
   :CLEAR 1000
   :MAXFILES=2
   :DEFSTR I,A-E
   :DEFINT P,L,R,X-Z
20 YZ$=CHR$(27)+"p"
   :YQ$=CHR$(27)+"q"
30 GOSUB 2000
40 RP=1
   :LP=1
   :ZP=1
   :LC=1
   :ON ERROR GOTO 70
50 GOTO 180
60 CLS
   :PRINT§40,"";
   :RETURN
70 IF ERR=52 OR ERR=55 THEN PRINT"Errore
   nel Nome del File"
   :GOSUB 2000
   :RESUME 280
80 PRINT"Errore ";ERR
   :GOSUB 2000
   :RESUME 40
90 GOSUB 2300
100 IF EOF(1) THEN GOSUB 2400
110 I=INPUT$(1,1)
    :IF I=CHR$(13) THEN GOSUB 2400
    :GOTO 100 ELSE IF I=CHR$(10)  THEN 100
120 IF I=" " AND ZP=0 AND RP=1 THEN 100
130 IF I="^" THEN 350
150 ZP=0
    :MID$(IO,RP,1)=I
    :IF I=" " THEN LP=RP
160 RP=RP+1
    :IF RP> LL THEN GOSUB 2410
170 GOTO 100
180 PRINT TAB(3);YZ$;" DEFINIZIONE PARAME
    TRI DI STAMPA ";YQ$
190 PRINT§80,"Marg.Sinistro:";
    :GOSUB 2100
    :XL=VAL(I)
200 PRINT§100,"Marg.Destro:";
    :GOSUB 2100
    :IF VAL(I)<=XL THEN XR=XL+60 ELSE XR
=VAL(I)
210 PRINT"Marg.dall'Alto:";
    :GOSUB 2100
```

Lezione 25

Simulazione di liste in Basic

Nella lezione precedente è stato presentato il concetto di strutture di dati dinamiche e in particolare si è introdotto il concetto di lista.
Solo i linguaggi più evoluti però ammettono una gestione dinamica della memoria con la possibilità di allocare e rilasciare aree di memoria ad hoc a seconda dei dati forniti in ciascuna esecuzione.
Nel caso degli altri linguaggi è solo possibile "simulare" la gestione di strutture dinamiche ricorrendo all'uso di strutture statiche. L'interesse di tale operazione è legato alla opportunità di realizzare strutture di dati "linked", rispetto alle caratteristiche del problema, poiché evidentemente si deve rinunciare alla caratteristica di dinamicità di allocazione.
Vediamo dunque come simulare una lista in BASIC, per la costruzione di un programma che effettui inserimenti ordinati di dati, come abbiamo visto nella lezione precedente.
La struttura che simuliamo è costituita di un insieme di elementi "linked" ciascuno dei quali è un record composto:
- del valore;
- del puntatore all'elemento successivo.

Poiché il BASIC non consente di generare dati dinamicamente ci appoggeremo a una struttura di array.
Poiché inoltre non possiamo costruire array di record, useremo due differenti array, rispettivamente per i valori da ordinare e per i puntatori. Nell'array dei valori inseriremo sequenzialmente i valori che ci sono forniti; per ciascuno dei valori forniti scriveremo nella corrispondente posizione dell'array dei puntatori l'indice dell'elemento immediatamente superiore.
Se quindi i valori inseriti nella lista sono i seguenti:

30 3 50 100 7

i due array si presenteranno così:

	VAL
1	30
2	3
3	50
4	100
5	7
6	
7	
8	

	PUN
1	3
2	5
3	4
4	−1
5	1
6	
7	
8	

e dovremo inoltre conoscere "l'indirizzo" del primo elemento della lista che in questo caso è 2. Infatti se partendo dalla seconda posizione dell'array dei valori:
- visualizziamo il valore in esso contenuto:

Completata questa venticinquesima lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi

EDITT.DO
EDITP. BA

contenuti nella cassetta "5 esercizi di programmazione", lato B.

I titoli seguiti dal suffisso DO corrispondono a testi, quelli seguiti da BA a programmi in BASIC.
Caricateli secondo le modalità che avete appreso.

• assumiamo come nuovo "puntatore" il valore dell'array dei puntatori che si trova nella posizione corrispondente:
• procediamo fino a trovare il puntatore "vuoto" (che abbiamo indicato con un -1); otterremo i suddetti valori ordinati in modo crescente.
Così, partendo dalla radice 2, troviamo VAL(2)=3 e proseguiamo con l'elemento in PUN(2), cioè con l'elemento di posizione 5; quindi troviamo VAL(5)=7 e proseguiamo con PUN(5)=1; ancora VAL(1)=30 e proseguiamo con PUN(1)=3; quindi VAL(3)=50 e proseguiamo con PUN(3)=4; infine, VAL(4)=100 e ci arrestiamo essendo PUN(4)=-1.

Costruzione della lista

Per costruire la lista dovremo:
• inserire i valori da ordinare nell'array dei valori;
• per ogni valore inserire:
••• nella posizione corrispondente dell'array dei puntatori l'indice dell'elemento immediatamente superiore;
••• l'indice del nuovo elemento nel puntatore dell'elemento precedente;
• costruire la radice della lista, cioè ricordare l'indice dell'elemento più piccolo per poter successivamente iniziare la scansione.
Cominciamo quindi a costruire lo "scheletro" della procedura di inserimento; innanzitutto creiamo la radice della lista, usando il primo elemento della coppia di array per il primo elemento della lista (procedure "crearadice") e indicando in NEXTIDX la prima posizione libera sulla coppia di array:

```
crea radice
NEXTIDX:=2;
WHILE NEXTIDX<=N DO
  BEGIN
      inseriscivalore;
      posiziona indici
      NEXTIDX:=NEXTIDX+1
  END;
```

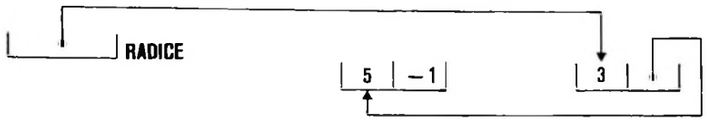
Inizialmente la lista è vuota, pertanto dobbiamo generarne la radice che sarà il primo elemento inserito; poniamo quindi il primo valore nella prima posizione dell'array dei valori e poniamo un valore di fine lista nella corrispondente posizione dell'array dei puntatori. Consideriamo inoltre che poiché al primo inserimento la lista contiene un solo elemento, questo rappresenta evidentemente la radice.
La generazione della radice sarà dunque la seguente:

```
V(1):=valore;
P(1):=-1;
R:=1;
```

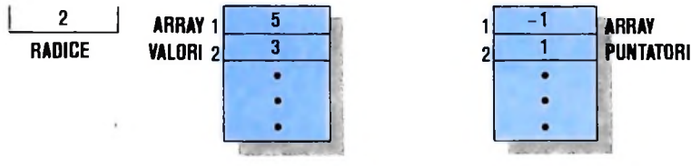
Si noti che è stato usato il valore -1 per indicare il puntatore "vuoto", cioè quello che corrisponde alla fine della lista.

Esaminiamo adesso il problema del posizionamento degli indici.
Cominciamo a considerare il caso in cui l'elemento da inserire sia il più piccolo di quelli già inseriti e vada quindi all'inizio della lista. Supponiamo per esempio di avere il valore 5 nell'array dei valori e di voler inserire il valore 3. La situazione iniziale è la seguente:

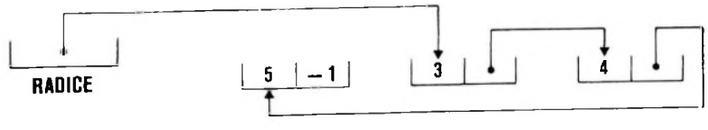
Dopo l'inserimento dovrà essere la seguente:



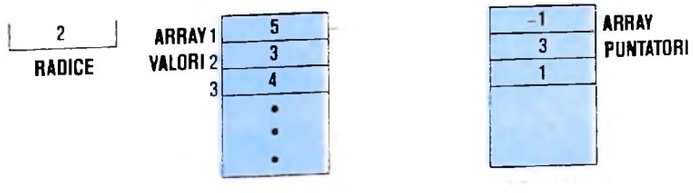
Se esaminiamo i corrispondenti valori nei due array troveremo:



Le operazioni da eseguire in questo caso sono dunque:
• il puntatore alla radice "punta" al nuovo elemento inserito;
• il puntatore dell'elemento inserito rimanda all'elemento che costituiva inizialmente la radice.
Consideriamo invece il caso in cui il nuovo elemento debba essere inserito, "logicamente", in una posizione intermedia. Inseriamo per esempio 4. La situazione dopo l'inserimento sarà la seguente:

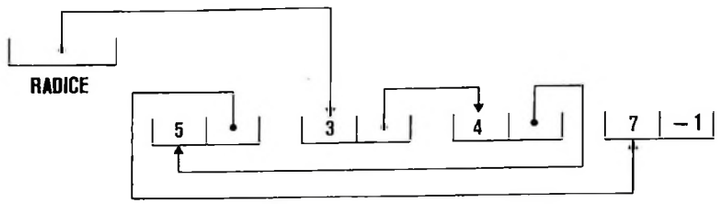


E la corrispondente situazione degli array sarà la seguente:

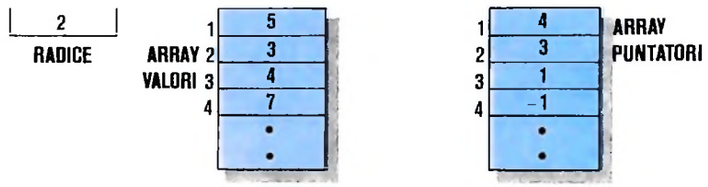


Come si può notare sono stati effettuati i seguenti cambiamenti:
- l'elemento precedente quello inserito "punta" al nuovo;
- il puntatore dell'elemento inserito "punta" all'elemento precedentemente puntato dal predecessore.

La radice e l'ultimo elemento della lista sono rimasti immutati.
Consideriamo infine il caso in cui l'elemento da inserire sia più grande di tutti quelli già presenti. È il caso in cui inseriamo 7.
La situazione che si produrrà è la seguente:



a cui corrisponde il seguente stato degli array:



È cioè cambiato l'ultimo elemento della lista; pertanto:
- il puntatore di quello che era ultimo "punta" ora all'ultimo inserito;
- il puntatore dell'ultimo inserito contiene l'indicatore di fine lista.

Riassumendo abbiamo individuato tre differenti casi:
- inserimento in testa alla lista;
- inserimento in posizione intermedia;
- inserimento in coda alla lista.

Il programma dovrà distinguere questi tre casi e trattarli opportunamente.

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- Come si realizza una lista in BASIC
- Le modalità di inserimento di dati in una lista in accordo alla posizione di inserimento.

```
    :FL=VAL(I):IF FL=0 THEN FL=4
220 PRINT"Righe per Pagina";
    :GOSUB 2100
    :PL=VAL(I)
    :IF PL=0 THEN PL=54
230 LL=XR-XL+1
    :IO=SPACE$(LL)
240 PRINT"Numerazione da Pagina:";
    :GOSUB 2100
    :XN=VAL(I)
250 PRINT"Intestazione: ";
    :GOSUB 2100
    :CT=I
260 DV="LPT:"
270 PRINT"Vuoi interlinea forzata ";
    :GOSUB 2100
    :IF I="S" OR I="s" THEN LF$=CHR$(10)
280 CLS
290 PRINT§5,YZ$;" DISPONI DEI SEGUENTI DOC
    UMENTI: ";YQ$
300 PRINT
310 FILES
    :GOSUB 2200
    :PRINT
320 PRINT§201,"NOME DEL FILE DA STAMPARE:";
    :INPUT I
    :CLS
330 OPEN "RAM:"+I FOR INPUT AS 1
340 OPEN DV$ FOR OUTPUT AS 2
    :GOTO 90
350 IF RP>1 THEN GOSUB 2400
    :ZP=0
360 I=INPUT$(1,1)
370 IF I="C" OR I="c" THEN LINE INPUT£1,I
    :PRINT£2,SPACE$((LL-LEN(I))/2+XL);I;LF$
    :LC=LC+1
    :GOTO 100
380 IF I="R" OR I="r" THEN LINEINPUT£1,I
    :PRINT£2,SPACE$(XR-LEN(I));I;LF$
    :LC=LC+1
    :GOTO 100
390 IF I="F" OR I="f" THEN GOSUB 2500
    :GOTO 100
400 GOTO 100
1990 '*********************************
1992 '*        routine delay          *
1994 '*********************************
2000 FOR X=2000 TO 1 STEP -1
     :NEXT X
     :RETURN
2090 '*********************************
2092 '*        routine input          *
2094 '*********************************
2100 I=""
     :INPUT I
     :RETURN
2200 I=INKEY$
     :IF I<>""THEN 2200 ELSE RETURN
2290 '*********************************
2292 '*    routine marginatura alta   *
2294 '*********************************
2300 FOR X=1 TO FL
     :PRINT£2," ";LF$
     :NEXTX
     :RETURN
2390 '*********************************
2392 '*   routine trattamento testo   *
2394 '*********************************
2400 LP=RP
     :ZP=1
2410 IF LC>PL THEN GOSUB 2500
2420 PRINT£2,SPACE$(XL);LEFT$(IO,LP-1);LF$
     :LC=LC+1
2430 W$=MID$(IO,LP,RP-LP+1)
     :MID$(IO,1,LL)=SPACE$(LL)
2440 IF LEN(W$)>0 AND LEFT$(W$,1)=" " TH
EN W$=RIGHT$(W$,LEN(W$)-1)
     :GOTO2440
2450 MID$(IO,1,LEN(W$))=W$
     :LP=1
     :RP=LEN(W$)+1
2460 IF EOF(1) THEN GOSUB 2500
     :CLOSE
     :MENU ELSE RETURN
2500 LC=1
     :PRINT£2,CHR$(12);
     :IF EOF(1)THEN RETURN ELSE GOSUB 2300
     :IF XN>ØTHEN XN=XN+1
     :PRINT£2,SPACE$(XL);CT;SPACE$(XR-XL
     -LEN(CT)-4);XN;LF$
     :PRINT£2," ";LF$
     :PRINT£2," ";LF$
     :LC=LC+3
2510 RETURN
```

Elaborazione di immagini

L'elaborazione di immagini mediante computer non è solo un'applicazione curiosa e interessante. Va diffondendosi solo da poco sugli elaboratori di piccola capacità, ma già da tempo è un campo di grande importanza, sui grandi elaboratori, per applicazioni come la fotografia astronomica (da terra, da satelliti o da veicoli spaziali), la fotografia meteorologica (da satelliti), o lo spionaggio. Ricostruire da un'immagine sfocata o poco nitida un'immagine precisa può essere di grande importanza per una organizzazione di "intelligence" (le esigenze militari e paramilitari hanno sempre costituito un fattore trainante nella ricerca informatica); per usi più pacifici, è importante poter ricevere dallo spazio informazioni in forma digitale e da queste ricostruire un'immagine, cui magari attribuire "falsi" colori che però evidenzino caratteristiche rilevanti. Queste ultime tecniche sono molto diffuse in tutti i settori del rilevamento, dove nelle immagini possono essere incorporate informazioni fornite da luce di lunghezze d'onda diverse da quelle visibili (infrarosse, X, gamma, e via dicendo) e che quindi non potrebbero essere presenti in una normale immagine "ottica".

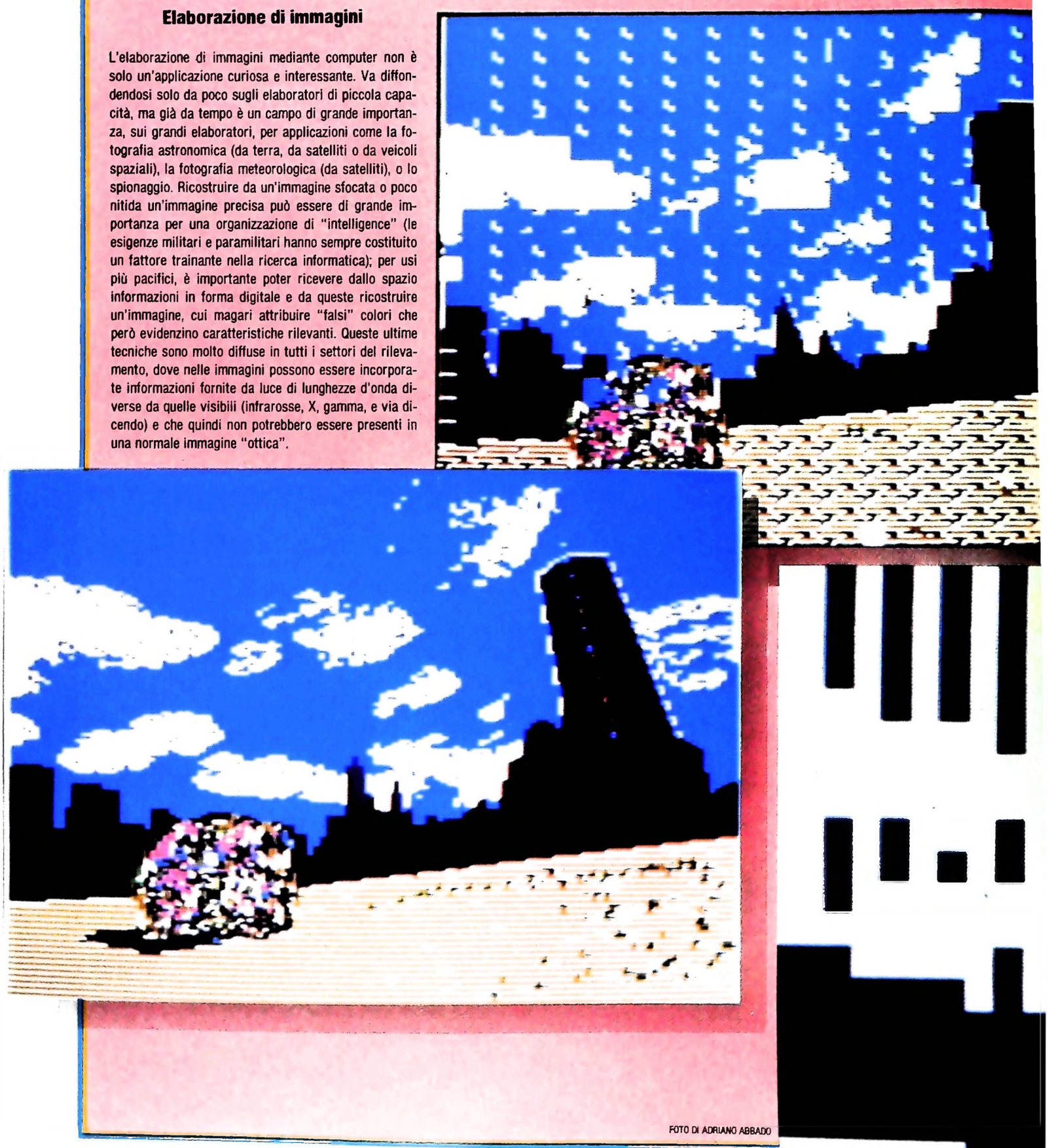
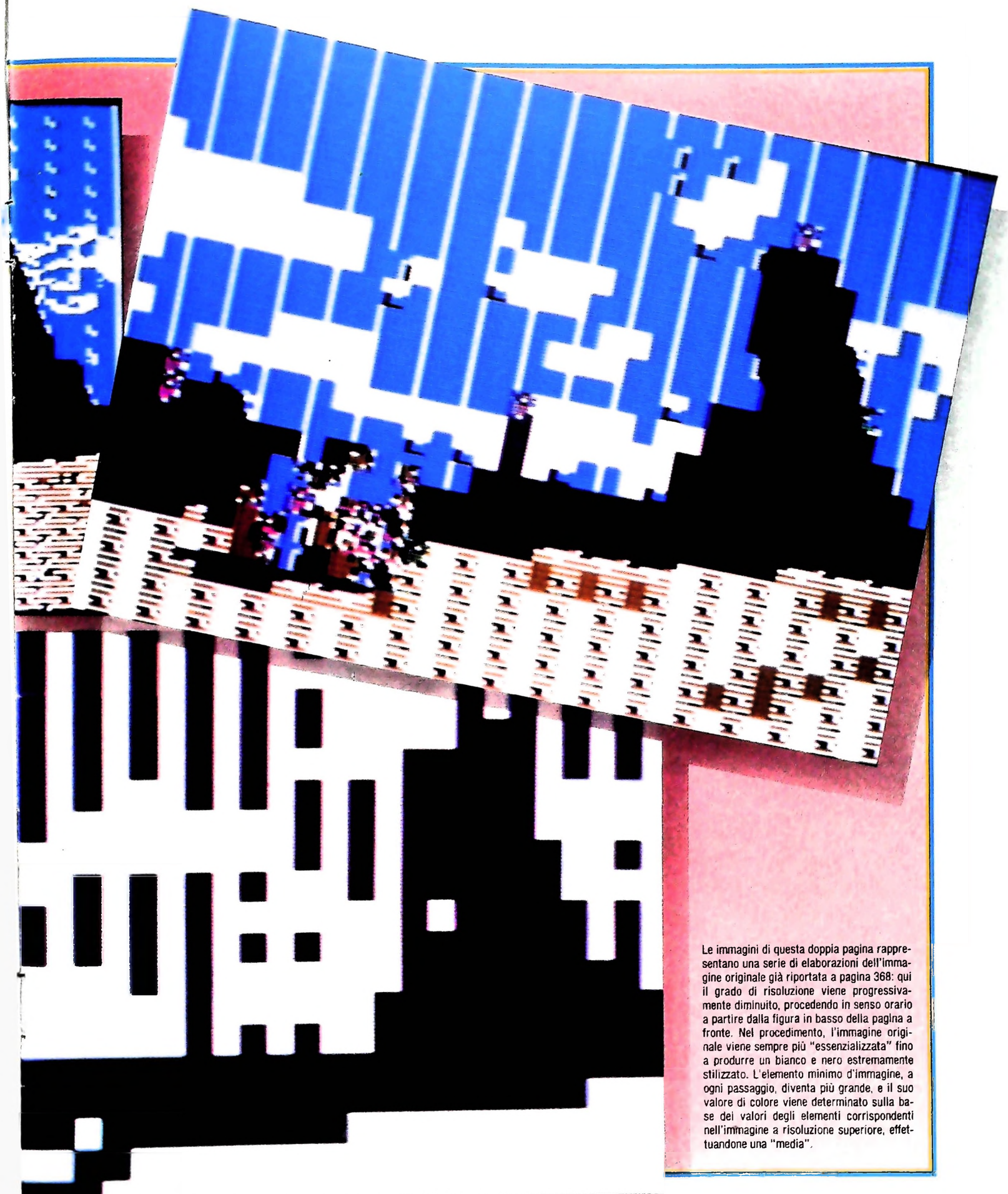


FOTO DI ADRIANO ABBADO



Le immagini di questa doppia pagina rappresentano una serie di elaborazioni dell'immagine originale già riportata a pagina 368: qui il grado di risoluzione viene progressivamente diminuito, procedendo in senso orario a partire dalla figura in basso della pagina a fronte. Nel procedimento, l'immagine originale viene sempre più "essenzializzata" fino a produrre un bianco e nero estremamente stilizzato. L'elemento minimo d'immagine, a ogni passaggio, diventa più grande, e il suo valore di colore viene determinato sulla base dei valori degli elementi corrispondenti nell'immagine a risoluzione superiore, effettuandone una "media".

VALUTARE UN LINGUAGGIO DI PROGRAMMAZIONE

Chiarezza e leggibilità, strutture di controllo, facilità e velocità di compilazione sono alcune delle caratteristiche essenziali di un linguaggio di programmazione.

Ci sono diversi punti di vista da cui possiamo guardare un programma per valutare le caratteristiche essenziali che un buon linguaggio di programmazione dovrebbe ragionevolmente avere: possiamo infatti guardare gli aspetti di *facilità di costruzione dei programmi*, oppure, da un punto di vista più prettamente industriale, i *costi di costruzione e di uso dei programmi*.

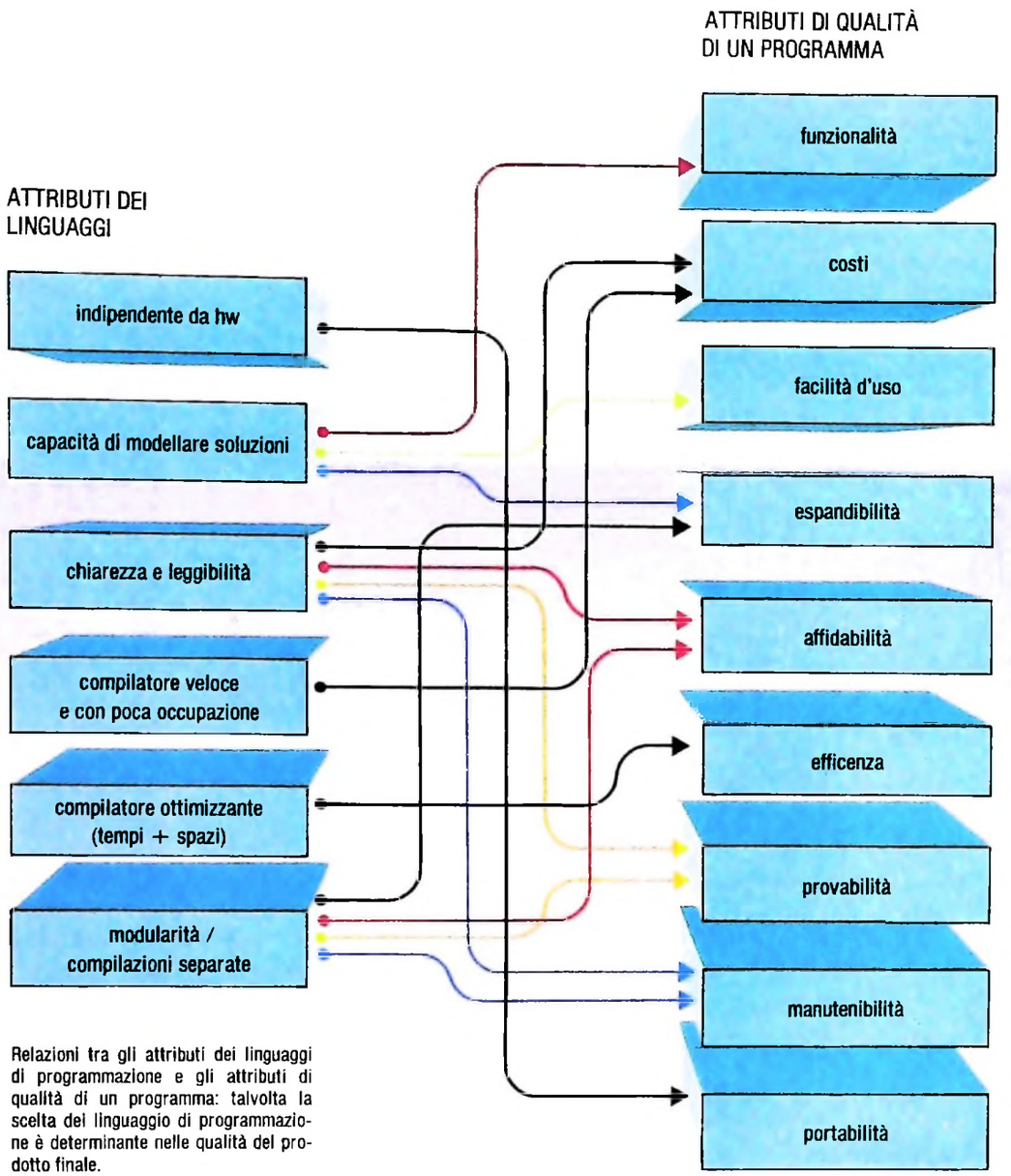
Il punto di vista del programmatore

Un linguaggio di programmazione non è solo uno strumento di comunicazione tra programmatore e calcolatore, ma è da considerare anche uno strumento di comunicazione tra un programmatore e un altro o addirittura tra un programmatore e se stesso.
Mentre è evidente il primo dei tre aspetti, esaminiamo brevemente i restanti.
Quando un programmatore ha completato un programma non ha in realtà concluso il suo sforzo: infatti ci si deve aspettare che il suo continuo uso, la riduzione dei problemi grazie ad esso, i cambiamenti nelle tecnologie dell'hardware e nuove esigenze che possono sorgere faranno "invecchiare" rapidamente il suo programma, che dovrà quindi essere modificato, esteso, aggiornato per poter fare fronte a tutte le nuove richieste. Nell'ambiente di produzione industriale di software i programmatori non sono stabili e immutabili, ma, come in tutte le aziende, crescono, cambiano i loro ruoli e i loro compiti, migrano verso altre società, e così via; quindi un programma è un bene dell'azienda che deve essere modificato e fatto crescere da programmatori che non necessariamente sono coloro che hanno inizialmente costruito la prima versione. Quindi è necessario che questi ultimi siano completamente in grado di capire il modo di funzionare del programma, le motivazioni delle scelte, l'algoritmo sottostante, e così via. In ogni caso, questa forma di comunicazione da un programmatore a un altro vale anche nel caso che sia lo stesso sviluppatore iniziale a mantenere in vita il suo programma: è infatti largamente verificato che anche colui che ha scritto il programma, dopo qualche mese, rischia di non ricordare le scelte fatte e di non comprendere il modo di funzionamento del suo prodotto (ecco perché sono così importanti i commenti!).
Inoltre, chi costruisce un programma o legge un programma altrui e lo comprende, impara anche a conoscere le caratteristiche del problema che il programma è chiamato a risolvere, e acquisisce una precisa esperienza nell'ambito della capacità di risolvere problemi.
Di più, guardando un programma come un "bene" prodotto, sarebbe opportuno garantire che la sua utilizzabilità sia la più ampia possibile, indipendentemente dal tipo di calcolatore che noi usiamo.
Quindi, un buon linguaggio di programmazione deve essere dotato delle seguenti caratteristiche:

- definizione non ambigua
- definizione indipendente dall'hardware
- capacità di modellare soluzioni
- chiarezza e leggibilità.

Definizione non ambigua e indipendente dall'hardware

La definizione di un buon linguaggio di programmazione deve essere fatta mediante strumenti formali (*grammatiche* o *sintassi*) che garantiscano che la sua forma sintattica e il suo significato siano interpretabili univocamente; inoltre un linguaggio di programmazione dovrebbe "nascondere" le specifiche caratteristiche hardware dei calcolatori sui quali è disponibile.
Se tali attributi sono verificati, un programma scritto in quel linguaggio di programmazione può essere trasportato su qualunque calcolatore, con la garanzia di mantenere inalterato il suo funzionamento.
Organismi internazionali come l'ISO e l'ANSI si occupano spesso di fornire definizioni standard di linguaggi di programmazione, a cui i costruttori di compilatori dovrebbero



Relazioni tra gli attributi dei linguaggi di programmazione e gli attributi di qualità di un programma: talvolta la scelta del linguaggio di programmazione è determinante nelle qualità del prodotto finale.

attenersi; di fatto tali standard vengono seguiti per lo più come linee guida, e ogni linguaggio di programmazione è accompagnato da un manuale che ne specifica le differenze dello standard, in termini di funzioni non disponibili, di estensioni e di diversi comportamenti. Ciò impedisce la migrazione immediata di programmi scritti in un certo linguaggio da un calcolatore a un altro; eccezione a questa prassi è il linguaggio ADA, che, definito e sviluppato per conto del Dipartimento della Difesa degli Stati Uniti, non accetta, per potersi chiamare con tale nome, alcuna differenza (nemmeno come estensioni!) dalla definizione ufficiale.

Capacità di modellare soluzioni

Un programma, come sappiamo è il risultato di un accurato processo di analisi e di soluzione di un problema: è quindi importante che un linguaggio di programmazione metta a disposizione il maggior numero possibile di strumenti concettuali che favoriscano il processo di analisi da parte del programmatore. Sappiamo che tale processo si basa in genere su meccanismi di *astrazione*, *classificazione* e *decomposizione*. Quindi è estremamente opportuno che un linguaggio di programmazione permetta:

Glossario

Accesso, metodo di - *Il metodo utilizzato dal sistema per accedere a un dato registrato in precedenza in memoria magnetica, o per immagazzinarlo in memoria magnetica.*

Accesso casuale - *Metodo di accesso alla memoria in cui recupero e immagazzinamento delle informazioni sono fisicamente indipendenti dalla posizione delle informazioni precedenti e seguenti. È il metodo usato normalmente per i dischi magnetici.*

Acceso diretto - *Metodo di accesso in cui recupero e immagazzinamento delle informazioni sono basati sull'indirizzo fisico della memoria, che coincide con una determinata posizione del braccio o della testina di lettura/scrittura.*

Accesso sequenziale - *Metodo di accesso in cui recupero e immagazzinamento delle informazioni avvengono per blocchi successivi: in questo caso la posizione di una informazione è strettamente dipendente dalla posizione delle informazioni precedenti e successive. È il metodo utilizzato con i registratori a nastro.*

Assemblatore - *Un programma che traduce le istruzioni di un linguaggio di programmazione (detto linguaggio di assemblatore o linguaggio assembly) in istruzioni in linguaggio macchina, trasformando i codici simbolici delle operazioni nelle relative istruzioni proprie dell'elaboratore, assegnando indirizzi di memoria alle istruzioni.*

Assembly, linguaggio - *Un linguaggio di programmazione simbolico che viene tradotto in linguaggio macchina da un assemblatore, con un rapporto uno a uno fra istruzioni simboliche e istruzioni in codice di macchina. È un tipo di linguaggio di "basso livello", ancora molto vicino al linguaggio macchina, legato cioè alle caratteristiche fisiche del calcolatore su cui deve essere utilizzato e, di conseguenza, non trasportabile da una macchina all'altra.*

Benchmark - *Letteralmente, punto di riferimento. Un test (o una serie di test) per valutare le prestazioni di prodotti hardware o software in confronto o isolatamente. Non esistono tecniche di benchmark soddisfacenti per tutte le categorie di prodotti e per tutti gli aspetti dei prodotti: si tratta di un campo che attende ancora una valida sistemazione metodologica.*

C - *Un linguaggio di programmazione di alto livello sviluppato ai Bell Laboratories da Dennis Ritchie come linguaggio per la programmazione di sistema per il sistema operativo UNIX, sul minicalcolatore PDP 11/70 della Digital Equipment Corporation. Gli obiettivi nella costruzione del linguaggio erano l'ottimizzazione del tempo di esecuzione, delle dimensioni e dell'efficienza dei programmi. Eccellente nella programmazione di sistema, si è dimostrato valido anche come linguaggio di uso generale, ed è ora disponibile anche su calcolatori personali. È relativamente piccolo, compatto, e di facile apprendimento.*

IR - *Sigla di Instruction Register, registro delle istruzioni. È uno dei registri dell'unità centrale di elaborazione (CPU), nel quale viene depositata l'istruzione che deve essere eseguita a quel momento dalla CPU stessa. Può essere (a seconda delle macchine) a 4, 8, 12, 16 o 32 bit.*

MAR - *Sigla di Memory Address Register, registro dell'indirizzo di memoria. È uno dei registri dell'unità centrale di elaborazione (CPU), nel quale viene depositato l'indirizzo della locazione di memoria prescelta, a quel dato momento, per una operazione di lettura o di scrittura.*

PC - *Sigla di Program Counter o contatore di programma. È uno dei registri dell'unità centrale di elaborazione (CPU), nel quale sono depositati gli indirizzi necessari per il controllo della successione delle istruzioni di un programma. Contiene normalmente l'indirizzo della locazione di memoria da cui dovrà essere caricata la successiva istruzione. Durante gli interrupt conserva l'indirizzo dell'istruzione.*

Plotter - *Un dispositivo di output di un sistema di elaborazione, che permette di ottenere disegni su carta e su altri tipi di supporto, come cartone e plastica. I plotter a penna disegnano tramite il movimento relativo di una o più penne su supporto fisso o mobile (plotter a tavola piana, plotter a rullo, plotter a tamburo). Nei plotter elettrostatici la carta passa sotto una testina (una serie di aghi) e viene sensibilizzata selettivamente dagli aghi elettricamente carichi; viene poi spruzzato un liquido contenente particelle di carbone in sospensione, che vengono attirate dai punti sensibilizzati e, infine, si ha un processo di essiccatura e eliminazione delle particelle in eccesso. I plotter elettrostatici non hanno parti meccaniche in movimento e risultano pertanto più veloci.*

RS-232 - *Un tipo di interfaccia standard per le comunicazioni tra apparecchiature di elaborazione. Si tratta di una interfaccia seriale (la trasmissione dei codici avviene cioè in serie, bit dopo bit) sviluppata dalla EIA (Electronic Industries Association) con la Bell System, produttori indipendenti di modem e società produttrici di calcolatori. Lo standard specifica un connettore a 25 pin e i segnali necessari per l'interfacciamento. Nonostante non sia l'unico tipo di interfaccia standard, è il più diffuso nel mondo dei calcolatori personali, per le comunicazioni fra calcolatori in via diretta o attraverso modem: molte macchine ne sono provviste di serie, per le altre è in genere disponibile come accessorio.*

SNOBOL 4 - *Un linguaggio di programmazione di alto livello sviluppato negli anni Sessanta ai Bell Laboratories, orientato alla manipolazione di stringhe di caratteri. Non possiede strutture a blocchi né dichiarazioni di variabili, ma dispone di molti strumenti per il riconoscimento, la generazione, la cancellazione e la sostituzione di stringhe e sottostringhe.*

Stampante - *Una periferica di uscita di un sistema di elaborazione, che consente la visualizzazione su carta. Esistono molti tipi di stampanti, che si raggruppano solitamente in due categorie: a impatto e non a impatto. Nelle stampanti a impatto il meccanismo di scrittura entra in contatto con la superficie del foglio, con un principio analogo a quello di una macchina per scrivere elettrica o mediante una serie di aghi che vanno a premere il nastro inchiostrato. Nelle stampanti non a impatto il meccanismo non entra in contatto con la superficie del foglio, ma crea il carattere con procedimenti termici (su carta opportunamente trattata chimicamente) o spruzzando inchiostro (stampanti a getto d'inchiostro) o ancora con procedimento simile a quello della riproduzione xerografica.*

Teletext - *Un servizio di informazione di tipo telematico, unidirezionale (dalla sorgente verso l'utente), inviato come parte del segnale televisivo. L'informazione può essere visualizzata su un normale televisore domestico, dotato di un opportuno decodificatore. L'informazione si presenta in "pagine", selezionabili mediante la pulsantiera di un telecomando, e può essere in quantità solo limitate. Il sistema può servire solamente a trasmettere informazioni di utilità generale come notizie di attualità, notizie meteorologiche, quotazioni di borsa, notizie di carattere locale. Il servizio è in fase di sperimentazione anche in Italia, con il nome di "Televideo", da parte della RAI.*

1. definizione di tipi di dati (questo permette infatti di costruire modelli "astratti" delle entità del problema)
2. strutture di controllo della programmazione strutturata
3. strumenti per lo sviluppo top down (per esempio, sottoprogrammi, o in genere capacità di organizzare un programma in moduli).

Leggibilità

A parte la banale osservazione che deve essere possibile arricchire il testo di un programma con commenti, è necessario che le frasi del linguaggio abbiano un "aspetto" spontaneo, espressivo e naturale per il programmatore (per esempio si pensi alla coppia FOR-NEXT come esempio di espressività, rispetto all'uso di istruzioni di tipo IF...GOTO per realizzare la stessa struttura).

Altri aspetti

Va da sé, inoltre, che un linguaggio dovrà permettere facilmente di usare qualunque tipo di apparecchiatura periferica (stampanti, terminali, dischi, cassette ecc.), e che potrà essere rilevante avere classi di linguaggi di programmazione particolarmente orientate a specifiche classi di problemi (linguaggi orientati al calcolo, alla grafica, all'elaborazione di grandi quantità di dati, al controllo di processi industriali ecc.).

Il punto di vista dei costi

Un linguaggio di programmazione richiede un compilatore; questo potrà produrre traduzioni dei programmi di partenza più o meno efficienti sia dal punto di vista del tempo necessario per l'esecuzione, sia dal punto di vista della quantità di memoria necessaria per far eseguire il programma.
Da qui emergono altri attributi che dobbiamo richiedere a un linguaggio:

- che sia facile da compilare
- che sia dotato di compilatori veloci
- che sia dotato di compilatori che occupano poca memoria
- che permetta di essere tradotto in modo da essere veloce nell'esecuzione
- che permetta di essere tradotto in modo da richiedere poca memoria per l'esecuzione.

Inoltre osserviamo che, quando si sviluppano grandi programmi, questi vengono scomposti in programmi più piccoli e in sottoprogrammi che potremmo pensare di compilare separatamente e collegare solo successivamente; ciò riduce automaticamente i problemi di tempi necessari per le compilazioni, in quanto, a fronte di una modifica o della correzione di un errore non si rivela necessario dover ricompilare l'intero grande programma, ma solo quella piccola parte che ha richiesto la modifica.

linguaggi	dimensione media (nº istruzioni sorgente)	produttività (nº linee/ mese uomo)	densità d'errore (nº errori/ istruzioni)	frequenza d'errori (nº errori/ mese uomo)	pagine di documentazione (documentazione/ istruzioni)
ASSEMBLER	30717	420	· 0266	3.4211	· 2727
FORTRAN	10722	321	· 0151	9.7514	· 0959
COBOL	27140	566	· 0129	· 7699	· 0679
PL/1	40767	391	· 0333	16.6586	· 1581
JOVIAL	39997	559	nn	nn	· 0993
PASCAL	103 000	589	nn	nn	nn
ALGOL	28952	530			
LEGENDA nn = non noto	Informazioni fornite da studi del Rome Air Force Development Center degli Stati Uniti mostrano come molti linguaggi di alto livello permettono di scrivere programmi grandi con elevata produttività e ridotta presenza di errori				

Quindi richiederemo anche la possibilità di effettuare *compilazioni separate.*
Ciò fa emergere naturale l'esigenza di un altro componente nella catena di costruzione dei programmi, di cui parleremo più diffusamente in seguito: si tratta del *linker*, cioè di un programma che, a partire da diverse porzioni di programma compilate separatamente, costruisce un unico programma scritto in linguaggio macchina direttamente caricabile in un calcolatore ed eseguibile.
Abbiamo posto molta enfasi sui compilatori, mentre abbiamo parlato poco di interpreti: in genere quest'ultimo tipo di programma è legato a piccoli calcolatori, comunque a quelli destinati a un largo mercato (si pensi proprio al linguaggio BASIC), dove le esigenze di facilità di uso e di apprendimento, nonché di scarsa occupazione di memoria sopravanzano quelle di velocità di esecuzione o di efficienza (non mancano comunque le eccezioni: per linguaggi come LISP o APL il discorso è un po' diverso).

Gli attributi dei linguaggi di programmazione

Tutte le richieste che sono state indicate perché un linguaggio di programmazione sia adeguato allo scopo non si ritrovano generalmente nella realtà dei fatti: spesso i linguaggi si presentano con differenti versioni non standard, talvolta non permettono di definire tipi di dati, oppure enfatizzano la modularizzazione rispetto alla leggibilità o l'efficienza dell'esecuzione dei programmi tradotti rispetto alla velocità del compilatore.
Nella carrellata sui linguaggi più significativi esamineremo via via le loro caratteristiche proprio secondo i punti di vista qui indicati.

Elementi di classificazione dei linguaggi di programmazione

Un linguaggio di programmazione è una notazione formale per la descrizione di algoritmi da eseguirsi su un calcolatore; come tale è descritto da:

- *sintassi* (che specifica come si scrivono i programmi, per esempio, la "carta sintattica" di un'istruzione di assegnamento)
- *semantica* (che specifica come si comportano i programmi; per esempio la descrizione degli effetti dell'esecuzione di un'istruzione di assegnamento).

I programmi "gestiscono" entità "passive" (le variabili che subiscono le trasformazioni) o entità "attive" (sottoprogrammi che, invocati, effettuano trasformazioni). Le caratteristiche di questi tipi di entità sono particolarmente rilevanti per la classificazione dei linguaggi.
Tali entità sono correlabili ad attributi (come il tipo di una variabile, o la sua posizione nella memoria del calcolatore), e l'associazione che viene effettuata dal linguaggio tra le varie entità e gli attributi è normalmente indicata con il termine inglese di *binding* (legame).
Il *binding* può essere definito prima dell'esecuzione del programma, al momento della sua stesura, e risultare in seguito immutabile (per esempio, la disposizione fisica in memoria delle variabili): si parla allora di *binding statico.*
Talvolta è invece possibile cambiare dinamicamente durante l'esecuzione del programma gli attributi delle entità (come distruggere variabili esistenti o metterne a disposizione di nuove): si parla in questo caso di *binding dinamico.*
Alcuni linguaggi di programmazione mettono a disposizione, per alcuni attributi, esclusivamente *binding statico*; altri permettono *binding dinamico.*

Le variabili

Una variabile è un'astrazione che corrisponde a un'area di memoria del calcolatore caratterizzata da un *nome* che viene usato come chiave d'accesso a un *valore.*
I principali attributi delle variabili sono:

- tipo (indica l'insieme dei possibili valori delle variabili e delle operazioni possibili)
- visibilità (specifica quali parti di un programma possono accedere alla variabile; in BASIC la visibilità di una variabile è totale da ogni punto di un programma)
- durata (specifica il momento in cui la variabile viene "creata" e quello in cui viene "distrutta"; per esempio, in un BASIC interpretato le variabili dimensionate vengono create al momento della dichiarazione DIM, e distrutte quando il programma viene distrutto).

Unità di programma

Un'unità di programma è un frammento (logico) di un programma, come un sottoprogramma BASIC, che permette di ridurre la complessità dei programmi e di procedere in modo top down.
In generale un'unità di programma può presentare le seguenti caratteristiche:

- visibilità (specifica da quali parti di programma un sottoprogramma è richiamabile; in BASIC, per esempio, un sottoprogramma è richiamabile da qualunque punto)
- disponibilità di variabili proprie (si tratta di avere variabili non visibili da altri sottoprogrammi; ha a che fare con la visibilità delle variabili)
- parametri, cioè possibilità di dichiarare, all'atto della chiamata, quali sono le variabili o i valori da elaborare, e non dover fare riferimento (come invece accade per il BASIC) a variabili fissate una volta per tutte (per esempio, le funzioni BASIC come SQR permettono di specificare un parametro, che corrisponde all'argomento).

Vedremo le caratteristiche suaccennate nei vari linguaggi.

UN NUOVO MODO DI USARE LA BANCA.

Conto corrente più

TANTI PENSIERI IN MENO CON IL CONTO CORRENTE "PIU'" DEL BANCO DI ROMA.

Essere cliente del Banco di Roma vuol dire anche essere titolari del conto corrente "più". Un conto corrente più rapido: perché già nella maggior parte delle nostre filiali trovate gli operatori di sportello che vi evitano le doppie file.

Più comodo, perché potete delegare a noi tutti i vostri pagamenti ricorrenti: dai mutui all'affitto, dalle utenze alle imposte.

Più pratico, perché consente l'utilizzo del sistema di prelievo automatico Bancomat e l'ottenimento della carta di credito.

Più esclusivo, perché potete usufruire del servizio Voxintesi, attraverso il quale chiedere direttamente al nostro elaboratore il saldo del vostro conto corrente con una semplice telefonata: in qualsiasi ora come in qualsiasi giorno, anche festivo.

Più sicuro, perché con una minima spesa potrete assicurarvi contro furti e scippi mentre vi recate in banca o ne uscite.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.