di Raffaello De Masi

V.I.P.

Il mio professore di latino alle scuole medie soleva ripetere durante le sue lezioni che molto spesso la facilità delle cose riesce a nascondere la loro complessità e potenza. Non a caso affermava che il più bel romanzo del mondo, «I promessi sposi» può essere letto con lo stesso piacere da un ragazzo di dodici anni come da una persona adulta, ovviamente con diverso frutto. Ma accanto a questa verità che oltre tutto mi pare lapalissiana, consigliava di non fidarsi, invece, di chi si nascondeva sotto forme e regole complesse. Per questo apprezzo il Basic ed il Prolog; ma ogni tanto trovo qualche perla che vale la pena di essere esaminata più da vicino. Se poi questa si presenta con caratteristiche inusitate merita una prova più accurata; da qui a scrivere di VIP il passo è stato veramente breve

Il pacchetto

VIP si presenta in una forma piuttosto scarna, costruito, com'è, da un grosso manuale redatto visibilmente con un pacchetto di DTP (pare proprio PageMaker) in carattere Bookman così ben leggibile. Il manuale di circa 250 pagine è accompagnato da due dischetti da 800K, anch'essi dotati di una scarna etichetta costruita con una Laserwriter, e da alcuni sottomano, rappresentati da cartoncini che riassumono le parole chiave e/o gli shortcut più utili e essenziali.

I due dischetti contengono, il primo, il system folder, praticamente inutile, e il linguaggio («pesante» circa 250K), oltre a un programmino di prova del genere «try me», che non fa altro che disegnare sullo schermo il logo di VIP, rappresentato da un cubo con le tre facce visibili contenenti le tre lettere del logo stesso. Il secondo dischetto, invece, contiene una serie innumerevole di interessantissimi e validi programmi che, esplorati, valgono meglio di qualunque tutorial nel guidare attraverso le varie fasi di programmazione. Tra questi programmi non mancano classici, come «La torre di Hanoi» o le diverse procedure di sorting; ma avremo modo di discuterne successivamente.

Cosa è e a che cosa serve VIP

Rispondere a questa domanda è facile e difficile; VIP, in parole piuttosto scarne, potrebbe essere descritto come uno dei tanti linguaggi già esistenti nell'area Mac; ma non è così semplice da definire. La filosofia di VIP (ricordiamo che VIP vale per Visual Interactive Programming) è quella di adottare una diversa tecnica di rappresentazione e di realizzazione di tutte le fasi della programmazione e della redazione del codice, riunendole in una struttura coerente e facile da leggere e interpretare.

Il principio animatore della filosofia di VIP è quello che il programmatore va aiutato e supportato in tutte le fasi del progetto, dal disegno di base fino al debug. VIP monitorizza continuamente l'operato del programmatore e interagisce con esso assicurando in ogni momento aiuto e difesa dall'errore. Esso si basa sulla premessa che una tecnica di programmazione ben strutturata offre notevoli vantaggi che si traducono in una chiara organizzazione e logica di programma. I realizzatori di VIP hanno integrato queste necessità con un editor del tutto inusitato, originale e raffinatissimo, che assicura una struttura corretta di ogni programma e previene la maggior parte degli errori in cui si può incorrere attraverso un editor più tradizionale.

Il primo passo di questa filosofia è basato sulla certezza che è più facile imparare e, comunque, realizzare un progetto, manipolando disegni che non testo. I costruttori del linguaggio dicono a chiare lettere, alla pagina 1 del manuale, che per ben programmare MAC, o si leggono attentamente tutti i capitoli di tutti i 5 o 6 volumi di Inside Macintosh, o si adotta VIP, che di quanto sta scritto in questa bibbia può fare a meno, e vi «può far risparmiare l'agonia della lettura di 70 capitoli». A parte la boutade (ma le parole che ho citate sono proprio scritte nel manuale), è vero merito degli autori di aver, per la prima volta dopo almeno una decina d'anni, proposto una nuova filosofia di redazione del codice sorgente che, e qui sta la novità, è né più né meno che una flowchart, ancorché di forma e tipo non convenzionale.

Le caratteristiche di VIP possono essere riassunte:

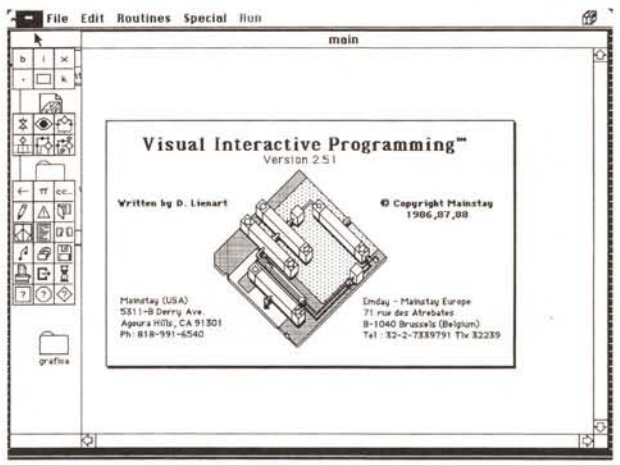
— VIP rappresenta un nuovo e mai tentato approccio alla programmazione che offre un livello di chiarezza e di ordine mai trovato anche nei più strombazzati esempi d'ordine e rigore. Esso, più che un linguaggio, è un ambiente che facilita e integra tutte le fasi della programmazione, dal disegno generale della struttura alla codifica al debug (che virtualmente, qui è quasi superfluo).

— VIP offre un aiuto concreto al programmatore dal primo momento fino alla fase di messa a punto finale. Ma non si tratta qui di un aiuto fittizio, come generalmente avviene attraverso il solito help in linea che poco o niente fa. VIP monitorizza continuamente tutto quanto realizzato dal programmatore, e

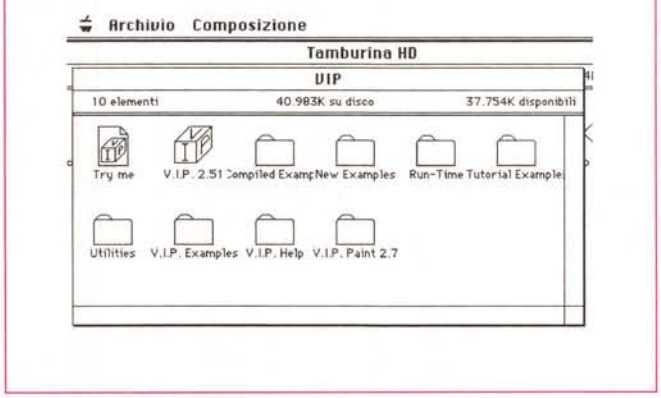
V.I.P.

Produttore:
Visual Interactive Programming for Apple Macintosh II MainStay 5311-B Derry Avenue Agura Hills CA 91301 USA.
Versione 1.0
Distributore:
Elcom - Via degli Arcadi, 2 - 34170 Gorizia
Prezzo: *L. 320.000 + IVA (9%)*

La finestra di apertura e presentazione del programma.



Il contenuto dei dischetti.



interagisce continuamente con l'utente, contribuendo in ogni momento a prevenire gli errori. In questo, come vedremo, è di grande aiuto lo speciale editor, che, in ogni momento, controlla la corretta struttura del programma.

Partendo dal principio indiscutibile (e già citato) che un disegno dice molto di più ed è molto più chiaro di qualsiasi scritto, la redazione del programma è realizzata solo attraverso l'uso di tool grafici. Il testo (e la tastiera) sono quindi relegati solo a semplice ruolo di manipolazione delle stringhe e, infatti, chi adotterà questo idioma dimenticherà presto la tortura delle fasi di input e di output comandate da ordini introdotti dalla tastiera proprie di tutti gli altri linguaggi. Tanto per capirci, per creare, ad esempio un loop o un salto condizionato, è sufficiente selezionare dalla palette relativa il tool caratteristico della struttura desiderata e, eventualmente, introdurre le variabili maneggiate. È tutto; il linguaggio si preoccupa di adeguare acconciamente la flowchart al tutto.

Un altro grande vantaggio di VIP è la sua perfetta integrazione con quasi tutte le chiamate al ToolBox, oltre tutto in un modo estremamente facile da realizzare. Per entrare in argomento, all'apertura lo schermo si suddivide in 3 palette allineate sul fianco sinistro e da una window di editing sul lato destro. Le procedure VIP sono suddivise in quindici classi, ognuna rappresentata da una icona sulla palette, che finalmente liberano il programmatore dalle tediose operazioni di debug grammaticale delle linee di programma. Questa semplificazione in sole quindici categorie non deve però trarre in inganno sulla potenza del linguaggio stesso; molte di queste procedure sono gerarchizzate e, anche a un semplice esame, mostrano una notevole potenza, essendo il risultato di librerie molto sofisticate e di routine ben costruite e accuratamente ottimizzate che assicurano una veloce esecuzione del programma.

La programmazione con VIP

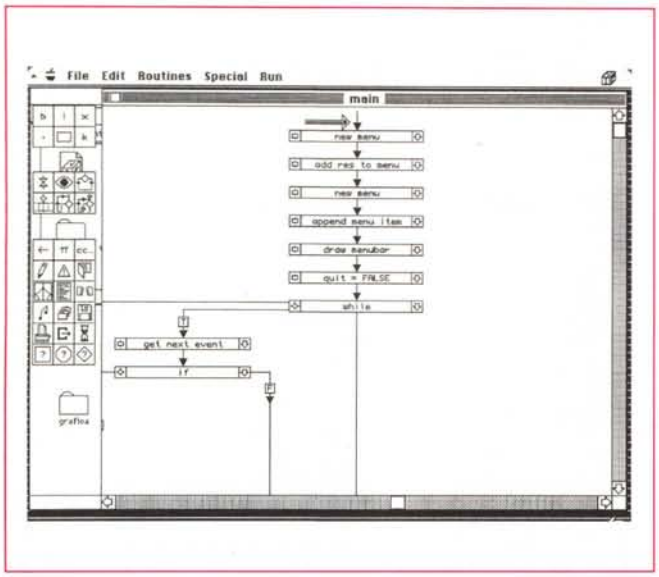
La programmazione con questo nuovo linguaggio si basa sull'utilizzo di tre grandi categorie di tool; gli oggetti, le procedure, e il toolbox. Essi sono i mattoni di base della programmazione; di queste tre parti parleremo in maniera disarticolata, essendo tra l'altro del tutto diverse tra loro come funzione e caratteristiche.

1) Gli oggetti

Parlare di oggetti è, sotto un certo punto di vista, parlare di variabili; per definizione, in VIP gli oggetti sono parti del programma destinate a contenere dati che possono essere manipolati dal programma. Tutti, ad eccezione delle costanti, possono essere definiti in forma locale o globale, secondo l'accezione utilizzata anche in altri linguaggi.

Esistono sei forme principali di oggetti; come vedremo tra poco, nella loro elencazione, ne esistono alcuni che non hanno praticamente riscontro altrove; eccone, in breve, una disamina.

- Byte; oggetto della classica grandezza di 8 bit; esso è destinato a contenere caratteri (secondo il codice ASCII) o numeri interi tra −128 e 127;
- integer; scompare l'inutile notazione su due byte; qui l'intero è rappresentato su 4 byte e può assumere il ben noto valore tra −2147833648 e +2147833647;
- real (che qui prende il nome di



Un esempio di struttura di un programma; tutti i blocchi vengono introdotti automaticamente selezionando il tool adatto dalle palette laterali.

extended); è il numero in virgola mobile, qui della lunghezza di 10 byte, con una mantissa ampia 64 bit (19 cifre significative) e un esponente codificato in 15 bit che può variare da −4932 a +4932;

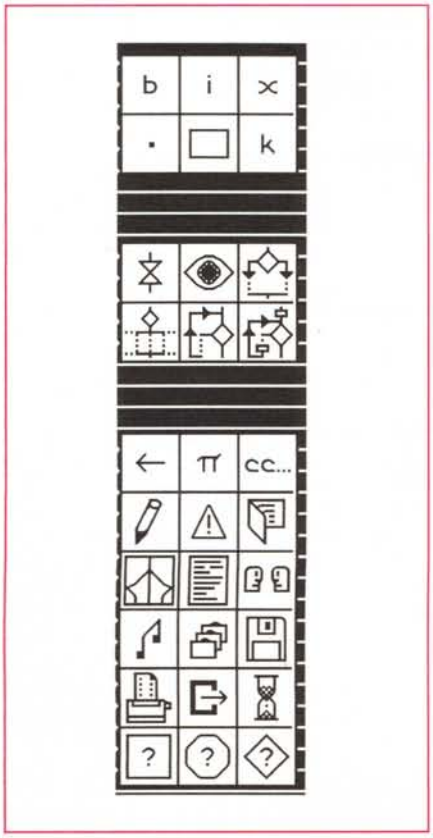
- point; qui andiamo sul nuovo, o almeno sull'inusuale, a confronto con gli altri linguaggi; si tratta di un oggetto ampio 4 byte, costituito da due numeri consecutivi (tra −32768 e +32767) che contiene la coppia di coordinate identificanti un punto sullo schermo;
- rectangle; la lunghezza è di 8 byte, la struttura è simile a quella dell'oggetto precedente, con la differenza che esso consiste di quattro interi indicanti le coordinate dei vertici estremi del rettangolo;
- costanti che, in analogia al loro nome e a quanto di analogo in altri linguaggi, permettono di rappresentare simbolicamente valori non destinati a variare nel tempo.

Gli oggetti, per la loro identificazione, seguono le stesse regole della definizione delle costanti e delle variabili negli altri idiomi; anche qui, comunque, VIP non tradisce la sua fama; infatti, cliccando la relativa icona, si apre una finestra di dialogo che permette di inserire il nome dell'oggetto stesso, di inizializzare il suo valore, di definire la sua globalità o località. Ma non basta; sfruttando la finestra di editing è possibile immediatamente definire array, che, per loro stessa definizione, sono serie di oggetti dello stesso tipo identificati da un nome unico. Non varia, in questo caso, la comoda definizione e individuazione degli array stessi attraverso la classica notazione nome-sequenza numerale. Sono permesse array a due e tre dimensioni.

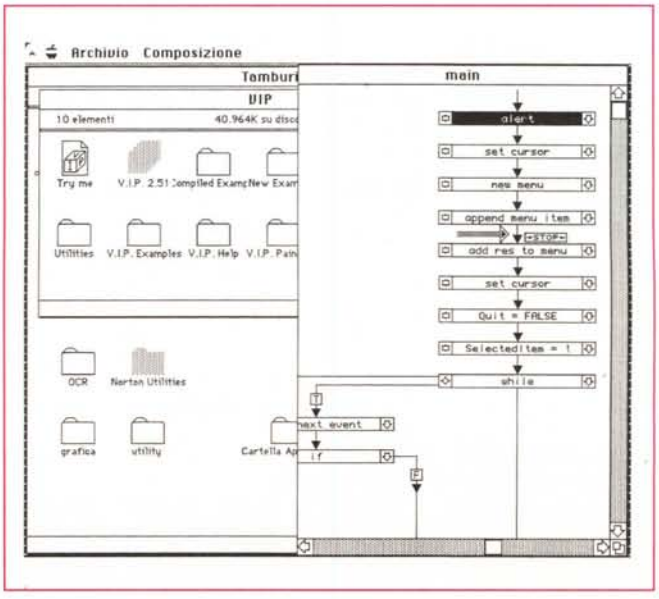
Gli oggetti e le loro relazioni con le routine

All'apertura, la finestra è rappresentata da quattro pezzi principali; la finestra di lavoro (che visualizza il programma vero e proprio, almeno a prima vista, del tutto inusuale), e tre finestre di tool, rispettivamente chiamate oggetti, forme logiche, e toolbox (ancorché differente da quello di ROM). È su queste tre entità principali e sulla loro rispettiva combinazione, che si basa la costruzione di un programma VIP.

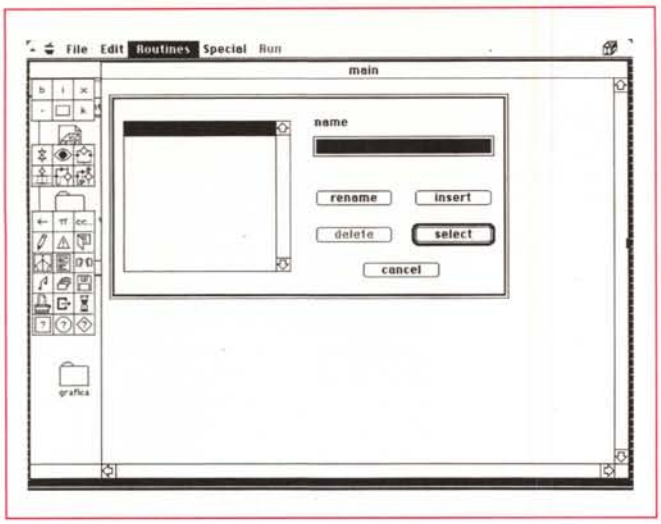
Il primo blocco contiene i sei oggetti già descritti, e non presenta grandi dif-

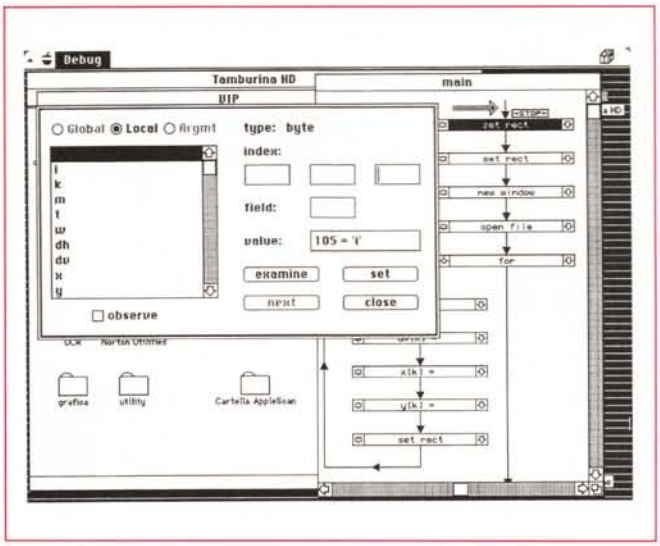


Le palette laterali, con i diversi modelli di parti del programma (v. testo).



Una fase di debug, con la possibilità di inserire breakpoint.





Un momento della definizione delle costanti e delle variabili.

ferenze con quanto esistente altrove. Nella definizione dei punti e dei rettangoli vengono in aiuto provvidenzialmente una serie di «edit field» presenti nella finestra di dialogo, che permettono di inserire ordinatamente i valori senza ricorrere a chilometriche rappresentazioni di valori concatenati, che, per la loro scarsa comprensibilità, oltre tutto cozzerebbero con la filosofia principale del linguaggio stesso.

Ma ritorniamo alla descrizione degli oggetti. Particolare interesse presenta il sesto tipo di oggetto, la Costante, che permette la definizione, come già detto, di costanti simboliche. Esse possono essere di quattro tipi fondamentali; carattere, intero, reale, e stringa; in quest'ultimo caso la stringa può contenere il codice di «escape», che in questa configurazione è rappresentato dal backslash.

Il programma mette a disposizione immediatamente quattro costanti predefinite, TRUE, FALSE) E e PI, ma si tratta solo del primo mattone; col tempo e con la paglia... è possibile costruirsi una palette di costanti di uso comune.

Una parola sull'allocazione di memoria; tutti gli oggetti, in default, sono definiti globalmente e la loro allocazione in memoria è del tipo statico. La memoria destinata a questi oggetti è riservata all'inizio del programma e resta allocata per tutta l'esecuzione.

Un secondo modo di allocazione di memoria è definito automatico. Gli oggetti cui si riferiscono sono definiti localmente; occupano memoria solo al lancio della routine che li utilizza, e svaniscono nel nulla quando la routine viene conclusa. La loro esistenza è di breve durata; come si è detto la loro vita dura esattamente quanto la durata della routine che li incorpora; il loro valore, inoltre, è indeterminato, vale a dire che essi, in default, non possiedono alcuna forma, né possono essere inizializzati al di fuori della routine stessa. Il limite massimo di memoria dedicabile ad essi è di 32K; possono sembrare pochi, nel nostro attuale mondo di mega e gigabyte, ma proprio per essere gli oggetti locali di vita labile, risultano nella maggior parte dei casi più che sufficienti). Occorre, quindi, che, come ben sanno programmatori appena appena scaltriti, appena si prevede di usare spazi di memoria più grandi dell'usuale, si passi alla definizione globale (cosa sempre consigliabile, ad esempio, nel caso di array), dove all'allocazione statica della memoria non corrisponde un vero e proprio limite, se non quello fisico.

Una terza metodologia di allocazione di memoria è quella dinamica; gli oggetti sono creati da una procedura e cancellati (e dissolti) da un'altra prima della fine del programma stesso. Si tratta della forma più specializzata e potente di gestione della memoria, dove gli oggetti hanno una breve esistenza sotto il controllo del programmatore stesso. Un esempio di oggetti di questo tipo sono le finestre, i menu, le figure, ecc.

2) La programmazione grafica usando oggetti e routine

Visualmente, un programma scritto in VIP è rappresentato da un insieme di riquadrini-scatole legati tra loro da linee (che, poi, in effetti, sono le linee di flusso) che rappresentano visualmente la sequenza logica di flusso di esecuzione del programma. Il box è il blocco, il mattone elementare del programma, la più piccola unità manipolata dall'editor di struttura. I box sono tutti della stessa forma e dimensioni, ed hanno un simile aspetto, salvo poi a svilupparsi in subforme diverse. Essi sono essenzialmente di tre tipi ognuno identificato da una piccola icona (simile, in forma, alla struttura di flowchart che rappresenta) presente sul lato destro.

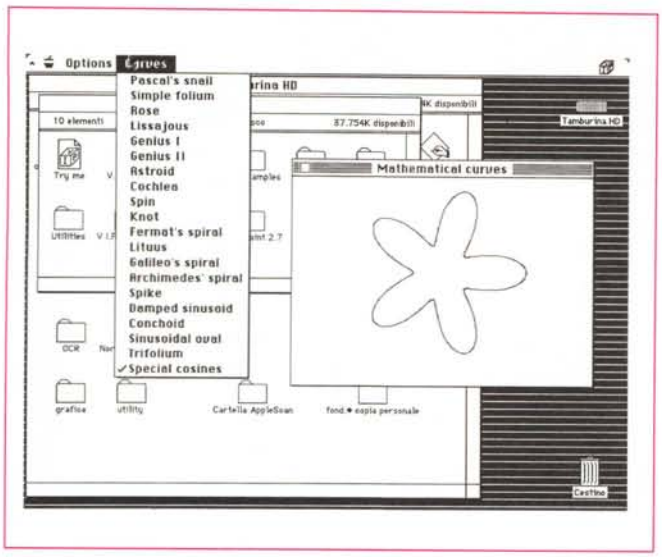
La prima delle forme possibili è la procedura, una istruzione per una predefinita operazione. Da essa esce, come è ovvio, una sola linea, che indica l'univoca prosecuzione del flusso di programma.

La seconda forma è il test che, come ovvio, valuta una espressione logica o aritmetica. In analogia con quanto già esistente in altri linguaggi, i tipi di test possibili sono i soliti if... then... else, while... do, for... next, e un inusuale switch.

La terza forma è la chiamata a routine, definita attraverso una precedente serie di ordini e istruzioni; un esempio di chiamata è «disegna_curva», «traccia_assi», o altro.

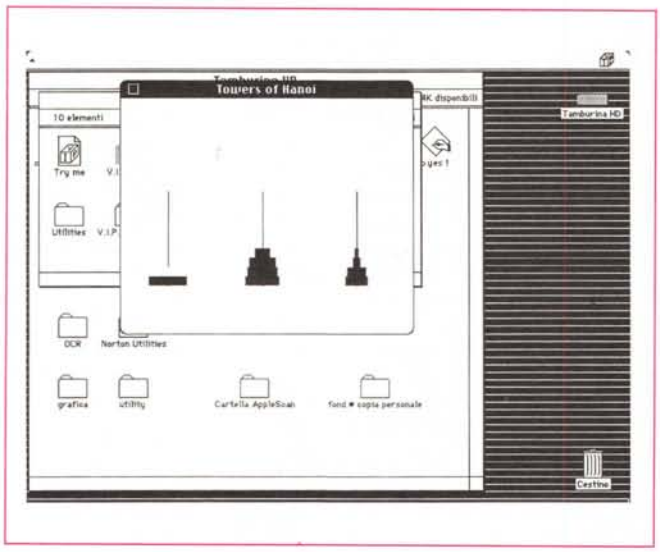
Attraverso la scelta, posizione e disposizione dei tre differenti tipi finora definiti è possibile costruire, virtualmente, la struttura completa del programma. Lo svilluppo di questo può essere regolato attraverso le usuali forme logiche, che di seguito esprimiamo:

- la sequenza; è la forma più semplice di passo di programma. In mancanza di altre definizioni l'editor assume essere la forma di default; proprio per questo



Un esempio delle funzioni ottenibili da un programma dimostrativo, incluso nel pacchetto.

Poteva mancare? Ecco la torre di Hanoi; compresa l'animazione sullo schermo, e utilizzando un FX, la torre di dieci dischi è stata risolta in meno di 9 secondi.



non esiste una specifica icona, ad essa riferita, nella palette delle forme logiche. Nelle figure che mostriamo, come poi generalmente avviene, lo sviluppo e l'ordine di esecuzione delle sequenze avviene dall'alto al basso.

- La struttura «if... then... else» che qui si arricchisce di un inedito «if not». Per questa forma strutturale esiste una apposita icona che, una volta selezionata, crea una struttura dotata di due braccia (le due possibili svolte del flusso, salvo poi a subramificare ancora il tutto) identificate da due icone del tipo T ed F, che portano, ovviamente, a diramazioni diverse del programma. Una forma più sofisticata di if... then è la forma switch (che strutturalmente assomiglia molto all'ON GOTO e ON GOSUB del Basic); essa consente il salto nel caso che le alternative siano più di due; il numero delle possibili uscite è, ancora una volta, definito da una specifica finestra di dialogo. Un vantaggio è rappresentato dal fatto che, dopo la definizione del numero di caselle da usare, esse sono inizialmente vuote, e non è necessario siano riempite tutte immediatamente. Addirittura il numero stesso delle caselle può essere aumentato o diminuito, in funzione della accresciuta o ridotta disponibilità delle alternative, anche dopo aver superato il noto su cui si sta lavorando (ad esempio in fase di debug o riscrittura del programma). Dimenticavo di dire che qui, come in tutti gli altri tipi di box, è possibile usare caselle per inserire commenti, vera chiave della comunicabilità di tutti i programmi.
- Altra struttura di chiara comprensione, attraverso l'agile disegno strutturale che immediatamente costruisce, è il costrutto while... do. Anche qui il tutto è realizzato mediante il passaggio attraverso un flag T, inizialmente vuoto; anche qui è previsto lo spazio per un commento (tutti i commenti, all'inizio, sono settati a ********).
- for... next è la terza form iterativa prevista; la sequenza di flusso è identica a while... do; l'intera struttura è regolata attraverso l'inizializzazione di quattro box, contenenti, come è noto, la variabile di controllo, il valore iniziale, quello finale e l'incremento; anche qui il box di T è settato inizialmente a zero.
- Chiamate a routine; si tratta, come è ovvio, della struttura più raffinata, complessa ed efficiente. Con molto acume il manuale definisce la routine come una idea, una struttura plasmata dal programmatore; la routine, con buona approssimazione, è la procedura del Pascal o la funzione del C; in altre parole è un sottoprogramma, dotato di membri, variabili e caratteristiche proprie; il vantaggio del nostro è che è possibile includere nel programma principale routine (o librerie di esse) precostruite, richiamandole direttamente attraverso il menu «File».

Creare una routine è molto semplice se si entra nella specifica filosofia «grafica» che sta alla base di questo tipo di programmazione; alla fatica talora improba della creazione di queste strutture in altri linguaggi si contrappone qui la tecnica di prelievo delle varie icone dalle palette; ma qui viene ancora di più a semplificare la cosa l'uso di un tool molto raffinato (rappresentato da un occhio spalancato); selezionando questa icona si ha un quadro complessivo della struttura della routine (ovviamente questo tool funziona anche su tutto il programma), su cui è possibile zoomare per vedere particolari ed eseguire un oculato debug.

Visto che ci troviamo a parlare di routine, è il caso di accennare ai tool a disposizione per costruirne di efficienti e funzionali (in effetti la soluzione migliore per realizzare un corretto ed efficace programma in VIP è di affidarsi, per quanto possibile, alle routine); VIP dispone di una serie di tool grafici di discreta potenza (si tratta di una serie di call a routine di toolbox e di QuickDraw), ma anche di funzioni matematiche raffinate (tra cui funzioni iperboliche) e di funzioni di stringa di discreto livello. Accanto a ciò esiste una gestione dell'errore articolata su 35 valori; non sono moltissimi, ma con un