500 Giochi

JACKSON LIBRI

500 Giochi
Angela Di Gregorio

Direttore Responsabile:
Giampietro Zanga

Direttore Editoriale
Roberto Pancaldi

Redattore di collana:
Cristiano Ruzza

Realizzazione Grafica:
Luca Ferrario

Impaginazione con tecniche di Desktop Publishing:
Publish Art di Fabio Bernareggi - via Aselli, 56 PAVIA

Fotolito e service di pellicolaggio:
C.S.G. srl - via Turati, 9/11 Grassobbio (BG)

Stampa:
LEGO-PRINT - Trento

Direzione e Redazione:
Gruppo Editoriale Futura S.r.l.
Via XXV Aprile, 39 - 20091 Bresso (MI)
Tel. 02/66526.1 - Fax 02/66526.222

JACKSON LIBRI

Indice

NOTE SULL'INSTALLAZIONE DI 500 GIOCHI

UTENTI WINDOWS 3.1X

1. Inserire il CD-ROM nell'unità CD
2. Avviare Windows
3. Aprire File Manager
4. Dal Menu File selezionare la voce **Esegui**
5. Digitare **X:JL500.EXE** dove **X** rappresenta l'unità del vostro CD, premere **INVIO**
6. Seguire le istruzioni a video

UTENTI WINDOWS 95

1. Inserire il CD-ROM nell'unità CD
2. Fare un clic sul pulsante Avvio e selezionare la voce **Esegui**
3. Digitare **X:JL500.EXE,** dove **X** rappresenta l'unità del vostro CD, premere **INVIO**
4. Seguire le istruzioni a video

SCHERMATA PRINCIPALE

Facendo clic su GIOCHI PER DOS passerete alla sezione DOS, i giochi sono divisi in tre generi, ARCADE, AVVENTURE e GIOCHI DA TAVOLO. Seguire attentamente a video le istruzioni.
Facendo clic su GIOCHI PER WINDOWS passerete alla sezione DOS, i giochi sono divisi in tre generi, ARCADE, AVVENTURE e GIOCHI DA TAVOLO. Seguire attentamente a video le istruzioni. A differenza dei giochi per DOS, quelli per Windows sono direttamente eseguibili dall'interfaccia grafica, con computer e lettori CD-ROM più lenti l'attesa sarà solo un po' più lunga, ma non abbiate paura i giochi partiranno in modo regolare.
Cliccando su ESCI oppure premendo in qualsiasi momento ESC uscirete dal programma.

CONSIGLI GENERALI

Dopo aver eseguito qualche gioco, riavviate pure l'interfaccia grafica e Windows, poiché spesso i giochi portano alla saturazione della memoria e non c'è niente di meglio che riavviare Windows per rimediare a questo problema.
Buon divertimento!

LA PROGRAMMAZIONE

UNA APPLICAZIONE È UNA MACCHINA

Un programmatore ha molto in comune con un ingegnere meccanico. L'ingegnere meccanico combina pezzi meccanici per costruire una macchina. Un programmatore costruisce programmi per calcolatore che offrono soluzioni a una classe di problemi. Un ingegnere meccanico utilizza ruote dentate, pistoni e bielle per costruire le sue macchine. Un programmatore usa istruzioni e formule — parole e numeri — per costruire i suoi programmi.

Il mondo dei programmi per calcolatore può essere diviso in software di sistema (che fa funzionare il calcolatore), programmi di utilità (che permettono di facilitare la gestione del sistema), e programmi applicativi (che permettono di eseguire lavori come l'analisi di uno stock azionario, la stesura di una lettera, o la creazione di una presentazione). Questo libro si concentra innanzitutto sui programmi applicativi.

Come del resto la precedente definizione suggerisce, un'applicazione è un programma sofisticato utilizzato in un ufficio per affrontare il lavoro quotidiano. Sono applicazioni, per esempio, i programmi di trattamento testi (come WordPerfect e Microsoft Word), i fogli elettronici (come Lotus 1-2-3 ed Excel), le basi di dati (come dBASE e Paradox), i programmi di analisi finanziaria (come CA-Simply Money e Quicken), le applicazioni grafiche (come PageMaker e CorelDRAW!), e i sistemi di contabilità (come QuickBooks e Microsoft Money). Ciò che rende unica un'applicazione all'interno di una categoria è l'uso che il programmatore dell'applicazione ha fatto degli strumenti disponibili e l'uso che l'utente dell'applicazione fa delle funzionalità offerte dall'applicazione.

Un'applicazione è come una macchina. Forse ricorderete dalle scuole superiori che una macchina esegue una quantità di lavoro proporzionale all'ammontare di energia di cui dispone. Per esempio, il motore di un'automobile consuma energia, nella forma di benzina, ed esegue del lavoro (muove la macchina). Un'applicazione funziona nello stesso modo: accetta dell'input dell'utente (pressione di tasti sulla tastiera) e produce lavoro (un documento di testo, un foglio elettronico, eccetera).

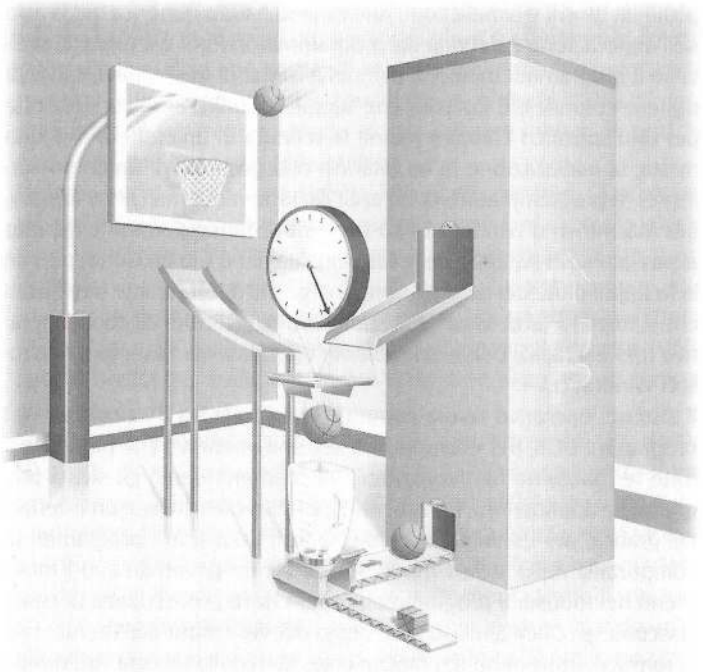
A sua volta simile a una macchina, il linguaggio utilizzato per scrivere un'applicazione ha dei componenti che funzionano insieme e che sono simili alle ruote dentate, pistoni e bielle. Quando vengono combinati nella maniera più opportuna questi componenti eseguono una quantità misurabile di lavoro. Il risultato dell'applicazione — il suo output — è direttamente collegato in modo proporzionale ai suoi dati

di input. Per esempio, un documento di testo è un prodotto di una serie di caratteri, una predizione basata sul foglio elettronico è un prodotto dei numeri che la compongono, e un rapporto è un prodotto di una base di dati.

UN'APPLICAZIONE ESEGUE DEL LAVORO

Se da bambini avete posseduto un meccano, ricorderete la gioia di combinare tra di loro ingranaggi, ruote e leve, applicando l'energia, e osservando come un assemblato funzionasse. La programmazione è un po' come il lavoro con il meccano: un programmatore combina dei componenti, ne scatena l'esecuzione e controlla come funzionano.
Per seguire l'esempio del meccano, il disegno di questa sezione mostra un gruppo di meccanismi che rappresentano come fluisce il lavoro all'interno di una moderna applicazione. L'applicazione ha inizio in alto a sinistra, dove l'utente digita dei dati (un numero) in una delle celle di un foglio elettronico. L'applicazione riceve quel dato nella forma di un evento dall'utente. Nel disegno, l'evento (rappresentato da una biglia rossa) fluisce all'interno di un tubo e viene lasciato cadere in una specie di ruota da luna-park che rappresenta un buffer. Buffer è un termine tecnico che indica un'area temporanea nella quale vengono memorizzati gli eventi sinché non sono pronti per essere elaborati. Gli eventi possono accadere con grande frequenza; vengono dunque salvati nel buffer e rilasciati a intervalli prestabiliti, determinati dalla velocità del calcolatore e prescelta dal programma.
Questa ruota (così come l'intera applicazione) è pilotata da una fonte di energia in rotazione, chiamata main. Il termine main, che in inglese significa "principale", è un altro termine tecnico usato nella maggior parte dei linguaggi di programmazione (specialmente il C e il C++). Si tratta semplicemente del componente principale del programma, che controlla quale dei componenti dell'applicazione è attivato in quale momento. Abbiamo scelto una grande ruota per rappresentare main perché il modulo main definisce il ciclo di base di un programma. Il modulo main dirige e controlla l'intera applicazione determinando quali routine vengono eseguite e in quali momenti. Le routine controllano i processi percepibili dall'utente, come la pulizia dello schermo, la lettura dei dati dalla tastiera o dal disco, e la scrittura di lettere o simboli sullo schermo. Il resto del disegno mostra gli ingranaggi e le leve che dipendono dalla fonte di energia principale, e rappresentano le routine che eseguono il lavoro vero e proprio sotto la spinta di main.

Dal buffer, la biglia evento fluisce all'interno di una "giostra" chiamata loop . Ancheloop è un termine tecnico, deriva dall'inglese, e significa "ciclo". Questo ingranaggio, insieme ai suoi simili etichettati calcoli e macro, costituisce gli elementi di elaborazione vera e propria — le formule e i gruppi di formule — che eseguono, come operai, il lavoro fisico.
La biglia dell'evento a questo punto viene spostata all'interno dello stack. Lostack, termine inglese che significa "catasta", rappresenta uno spazio in memoria. Nel nostro disegno, lo stack è rappresentato come un nastro trasportatore. L'evento viene trasportato nello stack per venire elaborato dai componenti logici del calcolatore, che sono rappresentati da un insieme di tubi che "magicamente" trasformano l'evento dell'utente in un prodotto (etichettato output) che il programma può effettivamente utilizzare. La "magia" che trasforma finalmente i dati nudi nell'output finale è determinata dalle formule e dalle macro degli ingranaggi dell'elaborazione dei numeri. La scatola rossa chiamata output rappresenta il prodotto finale. Per quanto riguarda il foglio elettronico, il contenuto della cella "Totali del trimestre" contiene questo output.



Facciamo un esempio pratico. Immaginiamo di aver creato un foglio elettronico che calcola le predizioni delle vendite annuali della vosta compagnia, basandosi sui rapporti settimanali della forza vendite. Dovete digitare i dati delle vendite settimanali (biglie eventi). Il buffer mantiene in memoria i dati delle vendite sinché il programma non è pronto ad elaborarli. I numeri che rappresentano le vendite settimanali vengono passati alla sezione di elaborazione numerica, dove vengono misurati, sommati, e ne viene calcolata la media. Il programma a questo punto piazza ciascuno dei numeri relativi alle vendite settimanali nello stack, sinché tutti i dati relativi alle vendite sono stati introdotti. Una volta che tutti i dati significativi vengono elaborati, la trasformazione dei dati può avvenire: e il risultato desiderato, cioè la previsione del fatturato annuo, viene restituito nella forma della scatola rossa.

MODELLO DELL'APPLICAZIONE

Un'applicazione, per come l'abbiamo definita, è uno specifico tipo di programma per calcolatore. Circa la metà dei programmi venduti oggigiorno rispettano la definizione di applicazione; l'altra metà è composta da giochi e simulazioni. Anche se la natura generica del lavoro dell'applicazione è definita dal programmatore (per esempio, la scelta se il programma manipoli dati nella forma di frasi e paragrafi o di righe e colonne), è l'utente che sceglie di utilizzare l'applicazione per uno specifico compito (come la scrittura di una lettera o di una rivista, la realizzazione di un bilancio o la gestione di un conto corrente). Il programmatore di un'applicazione moderna (per esempio, per Macintosh o Windows) può concentrarsi maggiormente rispetto al passato sullo sviluppo delle funzionalità per il suo prodotto, perché la maggior parte dei dettag li vengono gestiti direttamente dagli strati che si trovano "al di sotto" dell'applicazione — e cioè il sistema operativo (per esempio, DOS) e l'ambiente operativo (per esempio, Microsoft Windows).

Il sistema operativo lavora come un'interfaccia tra il calcolatore e i programmi DOS, per esempio, è il sistema operativo che permette a tutte le macchine IBM compatibili di "comprendere" gli stessi programmi e applicazioni. L'ambiente operativo offre invece un'interfaccia grafica; per esempio, Windows permette a tutti i programmi di comportarsi nello stesso modo — e di venire governati con i movimenti del mouse. Il programmatore non deve preoccuparsi di come ciascuna specifica applicazione possa ricevere input dall'utente, nella forma di movimenti del mouse o caratteri della tastiera, spedendo dati da e per il disco rigido, o scrivendo all'interno di una finestra;

sono DOS e Windows a offrire dei metodi predefiniti per fare queste cose.

Poiché anche DOS e Windows sono programmi, il calcolatore deve passare qualche tempo a eseguirli. Di conseguenza, il programmatore viene liberato dalla responsabilità di gestire personalmente i dettagli più minuti. Prima che Microsoft Windows divenisse l'ambiente operativo standard per i PC, tuttavia, un programmatore doveva concentrarsi anche su dettagli minuti come lo spostamento del cursore o il disegno delle finestre sullo schermo.

La struttura di un'applicazione è divisa in sei categorie, mostrate in questa pagina come fette di una torta. La torta è divisa in due metà, dove la metà di sinistra caratterizza le porzioni matematiche e di elaborazione dell'applicazione, mentre la metà destra si occupa dell'input e dell'output. L'altezza di ciascuna fetta della torta denota la sua importanza.

L'aritmetica (arithmetic) è certamente la fetta più importante sul lato sinistro. Un'applicazione, in genere, non può essere molto utile se non sa fare somme e sottrazioni. Persino un semplice editor di testi deve poter contare quanti caratteri fanno parte del suo documento.

La seconda più importante componente di un'applicazione è la manipolazione dei dati (data manipulation)— ivi compresa la capacità di salvare il lavoro in memoria e di recuperarlo; il trasferimento dei dati può avvenire verso uno spazio di lavoro temporaneo (la memoria) o lo stoccaggio a lungo termine (dischi floppy e dischi rigidi). Un programma non servirebbe a molto se non potesse salvare da qualche parte i risultati delle sue elaborazioni.

L'ultimo evento della parte sinistra è la Ricetta (recipe). Ciascun programma è composto da compiti repetitivi, come per esempio la ricerca di un documento specifico all'interno del disco rigido. Un cuoco potrebbe consultare il suo libro di cucina ogni volta che deve eseguire un compito molto frequente, come preparare un impasto o sbattere un bianco d'uovo a neve; analogamente, un programmatore utilizza una ricetta già studiata ogni volta che deve eseguire un compito comune.

Le Direttive (directions) sono la fetta più importante del lato destro. Le direttive sono i comandi che un utente dà all'applicazione — come per esempio la scelta di una voce di un menu all'interno di un foglio elettronico. L'introduzione di dati dalla tastiera (data entry), viceversa, consiste nell'invio di dati manipolabili all'applicazione. Un utente che scrive all'interno di un programma trattamento testi sta eseguendo l'introduzione di alcuni dati, e la stessa cosa avviene quando un'applicazione legge dei dati dal disco. Infine, la fetta display copre tutti i compiti di visualizzazione che spettano all'applicazione. Mostrare i

risultati dell'elaborazione fa parte di questa categoria, perché in teoria un'applicazione potrebbe eseguire un'elaborazione e non mostrare i suoi risultati all'utente; inoltre, l'utente non ha certo bisogno di vedere i risultati di ciascuna formula intermedia e di ciascun calcolo che un programma finanziario esegue — se lo facesse riuscirebbe solo a confondere il suo utente.

MA CHE COSA FA UN PROGRAMMATORE?

Lo scopo di un programmatore è di rappresentare una soluzione come un processo logico e matematico. Il programmatore si trova di fronte a una classe di problemi: per esempio, problemi matematici, come un sistema di contabilità, oppure problemi geometrici, come la disposizione di oggetti in uno spazio, oppure ricreazionali, come in un gioco. Il programmatore deve innanzitutto comprendere i principi e i metodi sottostanti a quel tipo di problemi e a questo punto decidere di quali elementi sarà composto il suo programma. A questo punto, il programmatore crea un modello logico matematico della soluzione universale per quella classe di problemi. In altre parole, il programmatore deve trovare un modo di descrivere tutte le parti della soluzione in una maniera logica.

Il calcolatore è una macchina logico-matematica, e pertanto lavora con numeri e formule. Per questo motivo, un programmatore trova relativamente semplice creare un'applicazione di contabilità, poiché la soluzione a un problema di contabilità deve manipolare numeri e formule. Se consideriamo, invece, un'applicazione che debba manipolare delle mappe, le cose sembrano più complesse. Compiti del genere possono sembrare tutt'altro che numerici — ma non dobbiamo dimenticare l'esistenza della geometria. Compiti come la decisione dove gli alberi e i cespugli vanno piantati per abbellire un edificio devono venire simulati con le istruzioni di un programma.

Prendiamo ad esempio un'applicazione creata per l'uso di alcuni medici, che generi una lista di diagnosi in risposta a una lista di sintomi. Una parte dell'applicazione manterrà liste dei sintomi, dei motivi e dei rimedi; questa parte sarebbe la più semplice da creare per il programmatore: si tratterrebbe di un programma di gestione di una base di dati (e si troverebbe all'interno della fetta "Manipolazione dati" della pagina precedente). Una base di dati è una collezione di dati strutturati (come l'elenco dei pazienti di un medico). Il sistema per la gestione della base dati è quel segmento dell'applicazione che controlla i contenuti della base dati (cioè le liste).

È certamente molto più difficile per il programmatore simulare un sistema che prende decisioni piuttosto che semplicemente mantenere le liste di sintomi e rimedi. A meno che il programmatore abbia una preparazione in medicina, dovrà eseguire ricerche estensive e consultarsi con i medici per comprendere appieno il processo tramite il quale un clinico decide come agire — un po' come se volesse diventare uno studente di medicina. In aggiunta alla conoscenza teorica, maturata all'università, un medico deve affidarsi in qualche modo a sensazioni e intuizioni quando diagnostica un rimedio. E un programmatore, dunque, dovrà creare un'applicazione che simuli l'intuizione del medico. Creare un programma che simula in tutto e per tutto l'intuizione del medico è probabilmente impossibile, e andrà dunque creata un'applicazione che simuli semplicemente le funzioni fattuali, lasciando l'intuito al medico che sta eseguendo l'applicazione. Quando un programmatore sviluppa del software che automatizza un compito normalmente svolto da una persona, quel programmatore deve comprendere compiutamente tutti gli aspetti del lavoro, e cioè i principi, le leggi fisiche, e la matematica che lo governa. Il programmatore non deve essere necessariamente un esperto del campo; per esempio, un programmatore che sviluppa un programma che giochi a scacchi non deve necessariamente essere un gran maestro del gioco.

GLI STRUMENTI DEL PROGRAMMATORE

L'utente di un progamma non deve necessariamente possedere sul disco rigido del suo calcolatore il linguaggio di programmazione che è stato usato per creare quell'applicazione.

I programmatori, solitamente, sanno usare più linguaggi di programmazione, e scelgono quello più adatto per scrivere un'applicazione che gli è stata commissionata, o spesso ne usano più d'uno contemporaneamente.

Un linguaggio di programmazione è semplicemente un intermediario tra il programmatore e la macchina. Il linguaggio di programmazione traduce le istruzioni del programmatore nel linguaggio nativo del calcolatore. Questo linguaggio nativo, sostanzialmente, è composto di cifre binarie (uno e zero), che sono di gran lunga più difficili per un programmatore umano da manipolare delle frasi che costituiscono il linguaggio di programmazione.

Un programmatore, dunque, scrive delle istruzioni testuali, collettivamente chiamate codice sorgente. Il codice consiste di istruzioni, cioè

elenchi di cose da fare — analoghe a una lista della spesa — che vengono eseguite in ordine. I linguaggi di programmazione che sono più vicini al linguaggio umano parlato, come l'inglese o l'italiano, sono chiamati linguaggi di alto livello. BASIC, Pascal, C, C++, Modula, Ada, PROLOG, COBOL e FORTRAN sono i linguaggi di alto livello più utilizzati oggigiorno. Il linguaggio macchina e il linguaggio assembly sono invece chiamati linguaggi di basso livello poiché sono più vicini alla natura fisica del calcolatore. Solitamente, quando si rappresenta come si è fatto un calcolatore (come noi abbiamo fatto nella pagina precedente), le funzioni vicine alla base del grafico sono quelle più vicine alla circuiteria del calcolatore, mentre quelle che si trovano più in alto sono più vicine al modo in cui le persone lavorano e colloquiano.

Un programmatore usa un compilatore per scrivere i suoi programmi. Un compilatore assomiglia vagamente a un programma di trattamento testi, e viene innanzitutto usato dal programmatore per scrivere le istruzioni che costituiscono il programma. Il compilatore, poi, traduce le istruzioni del programmatore in codice di basso livello che il calcolatore può comprendere ed eseguire — istruzioni chiamate codice oggetto. Il compilatore genera un file che contiene il codice oggetto dell'applicazione.

"IL PRIMO FOGLIO ELETTRONICO"

Dan Bricklin e Bob Frankston sono probabilmente i creatori del moderno modello di applicazione. Bricklin è l'inventore del foglio elettronico, mentre Frankston ha realizzato l'idea di Bricklin creando un prodotto, chiamato VisiCalc, effettivamente venduto sul mercato. L'originale applicazione di Bricklin, a differenza della maggior parte degli altri programmi che venivano sviluppati in quel periodo (1974), attendevano un evento da parte dell'utente prima di cominciare a elaborare i dati. Inoltre, con VisiCalc era l'utente a poter decidere a quale scopo venisse destinato il programma. Prima che Bricklin e Frankston presentassero il loro prodotto, il funzionamento di un programma era definito in anticipo dai suoi creatori. Se un programma, per esempio, serviva a calcolare la velocità media di un aereo in volo, non poteva venire usato per altro che per questo scopo. Bricklin, in origine, intendeva fare di VisiCalc semplicemente una macchina per calcolare e non un'applicazione. Frankston trasformò il concetto di "programma per il controllo della macchina" di Bricklin in software eseguibile, che molti considerano il primo programma applicativo moderno.

JACKSON LIBRI