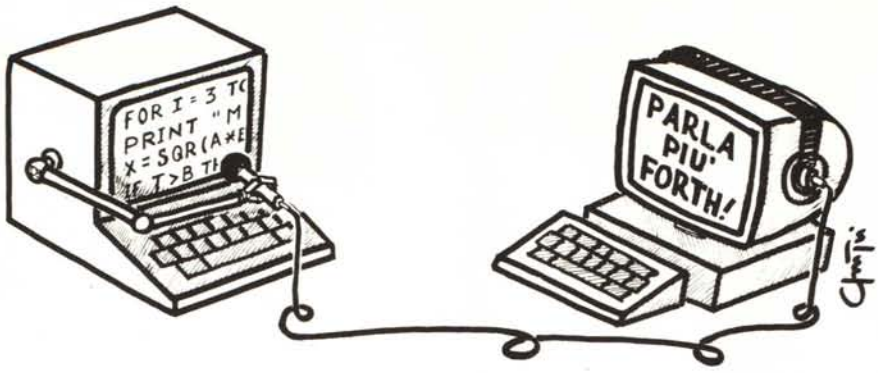

Parla più FORTH

di Raffaello De Masi

Ottava parte

Un po' di pratica

Cammina, cammina, Ermanno Olmi ci ha fatto un film e noi siamo arrivati a specializzarci ed a manipolare stack, flag e word quali esperti nocchieri (ve lo ricordate il nauta della prima declinazione?) nel procelloso mare del Forth. Ma poiché le calde e quiete lagune del Basic e del Fortran non ci avevano mai dato il brivido dell'avventura ci siamo spinti in mare aperto e ormai gli amici di una volta chi li vede più?

Ma il viaggio è stato duro e lungo e stavolta è il caso di fermarci. Per cui, per questa puntata, niente nuove word, ma qualche utile applicazione di quanto abbiamo fatto finora, con qualche programmino di utilità sempre buono da tener registrato da qualche parte.

Già parlammo, un paio di puntate fa, in occasione dei cicli DO-LOOP, delle word LEAVE ed EXIT. Ricorderemo che la prima determina la chiusura di un ciclo DO-LOOP, identificando il valore del contatore al limite del loop, mentre la seconda agisce come una vera e propria word di chiusura (quindi come un [;]) pur avendo la limitazione di non poter essere usata con un ciclo DO-LOOP, cui, comunque, assolve la già citata LEAVE.

Evidentemente sia LEAVE che EXIT vanno precedute da una comparazione, altrimenti non avrebbero motivo di esistere. Comunque è uso comune che EXIT sia generalmente utilizzato per routine di salvataggio, come, ad esempio, in caso di errore, mentre LEAVE interviene quando un evento desiderato avviene.

La figura 1 riporta un programmino che consente di ricercare, in uno specifico blocco di memoria, un certo numero e, in ossequio a quanto precedentemente affermato, usa LEAVE per uscire dal loop quando viene trovato il numero stesso. Per consentire ciò, l'indirizzo ricercato viene portato in stack, ma, se il numero non viene trovato in TOS, viene riportato a zero.

All'inizio, si assume, evidentemente, che il numero non sia stato trovato per cui viene posto in TOS uno zero. Stabiliti i limiti di DO-LOOP, il programma legge un numero in memoria e lo compara con il valore di test. Se i due valori sono uguali (ricerca ultimata) un IF-THEN sostituisce lo zero dello stack con l'indirizzo effettivo del numero ed esegue un LEAVE forzando il ciclo DO-LOOP. Altrimenti si riprende daccapo.

Al termine del ciclo sia esso stato fruttuoso o no, i due numeri in cima allo stack, ind 1 e n, vengono cancellati lasciando solo l'indirizzo ricercato, in caso di ricerca positiva, o lo zero, in caso negativo.

Come già abbiamo detto EXIT è generalmente utilizzato come escape da situazioni d'errore come, per esempio, un input non valido.

La definizione, data la puntata precedente, della radice quadrata può essere, in caso di errore quale la richiesta di essa per un numero negativo, così integrata (dopo la word iniziale)

```
...
DUP 0<
IF EXIT THEN
1
BEGIN
OVER OVER /
...
ecc.
```

Ricerca ed ordinamento in array e tabelle

In molte applicazioni, i numeri di una array rappresentano i risultati di test, raccolta informazioni o, cosa per cui il linguaggio è particolarmente tagliato, sequenze di dati ottenuti da un controllo di una macchina o di una strumentazione. Se tali informazioni debbono essere successivamente analizzate, può essere necessario il loro riordino sia in senso numerico (crescente o decrescente) od alfabetico. La figura 2 e successive descrivono due comuni tecniche di riordino (o sorting), generalmente le più usate, chiamate bubble sort ed insertion sort, e forniscono listati adatti alla loro implementazione.

Bubble sort

La tecnica di bubble sort è così chiamata poiché determina una continua risalita dei numeri in memoria, come bolle d'aria nell'acqua.

Ad esempio, in una array monodimensionale, ogni numero viene paragonato, partendo dal primo numero della lista stessa, con il successivo.

Se il numero risulta essere più grande del successivo, si effettua uno scambio tra di essi. I due numeri successivi vengono comparati, scambiati se necessario e così via. Dopo un certo numero di tentativi il programma avrà riordinato in senso crescente tutti i numeri ed il numero più grande sarà "ribollito" al posto più alto.

Questo algoritmo, universalmente conosciuto nel mondo dell'informatica ed applicato dappertutto in ogni linguaggio, richiede ovviamente diversi passaggi per completare il riordino. Purtroppo il numero di tali passaggi non è fisso, dipendendo esso dall'iniziale stato dell'array per cui il computer non avrebbe alcun termine di paragone per conoscere effettivamente quando interrompe l'operazione. È necessario, per tale ragione, utilizzare uno speciale indicatore che segnali se, in un passaggio, sia stato effettuato uno scambio. Tale indicatore, definito flag di scambio, avviserà il computer quando interrompere il sorting.

La figura 2 evidenzia la flowchart di bubble sort con uso del flag di scambio. Questo viene settato ad 1 all'inizio di ogni passaggio di sort. Viceversa ogni volta che un passaggio determina uno scambio di numeri il flag passa a 0. Se, pertanto, all'inizio di ogni ciclo di sort (o, che è la stessa cosa, alla fine della array) il flag presenta il valore 0, occorre ripetere l'operazione mentre per valore 1 (vero) di flag il sorting può essere ritenuto completo ed il programma esce dal loop.

Come si vedrà, l'operazione richiede almeno 1 sequenza di confronti (ipotesi minima per array già ordinata). Per una array messa nell'ordine esattamente inverso (caso più disastroso) occorrono (fatevi un pic-

colo conto od una prova con carta e matita) n-1 passaggi per il riordino totale +1 per la verifica finale. Vale a dire che, una array di N posti impiegherà da 1 ad N passaggi per il sorting con (N+1) /2 passaggi come media.

La figura 3 mostra la definizione della word BUBBLE, che consente appunto la bubble sort di una array di numeri (o di parole, tanto basta usare gli indicatori ASCII). Essa è facile da interpretare, seguendo precisamente la flowchart della figura 2. Notare lo uso di >R in coppia, che consente di salvare nel return Stack (lo ricordate, quanto tempo è passato!) i limiti di DO-LOOP, che rappresentano gli estremi della array in cui va effettuata la ricerca. Notare anche la sequenza [!!] che consente (il riordino dei parametri viene eseguito dal 4ROLL) di conservare il primo numero nel secondo indirizzo e viceversa, vale a dire consente lo scambio dei numeri se la condizione I 2 + @ I @ < (comparazione tra i due numeri) setta il flag a 1.

L'operazione eseguita dalla word BUBBLE ha però uno svantaggio. Ogni esecuzione del ciclo DO-LOOP causa il confronto di tutti i numeri in array, anche quelli che già sono stati scambiati e portati alla loro posizione finale. Poiché non c'è necessità di valutare ogni volta tali numeri, è possibile, ancora di più, accelerare l'operazione ammettendo, nell'ambito del programma, la comparazione di tali numeri. Vale a dire che, nel riordino dell'array, il primo passo analizzerà N numeri, il secondo N-1 numeri, il terzo N-2 e così via.

Pertanto, per definire una nuova word con un algoritmo più efficiente occorre modificare il programma precedente in modo da ridurre il limite di DO-LOOP di 2 ad ogni passaggio. La figura 4 mostra una versione migliorata della word BUBBLE, XBUBBLE in cui appunto il limite di DO-LOOP viene decrementato, ad ogni round, di 2, immediatamente dopo l'estrazione dal return stack. Inoltre l'indirizzo iniziale, per consentire effettivi n passaggi è incrementato di 2. Tutto il resto è uguale.

È davvero più veloce tale nuova definizione? Analogamente a quanto suggerisce Scanlon, abbiamo anche noi eseguito una prova su una array di 300 numeri messi esattamente al contrario (in senso decre-

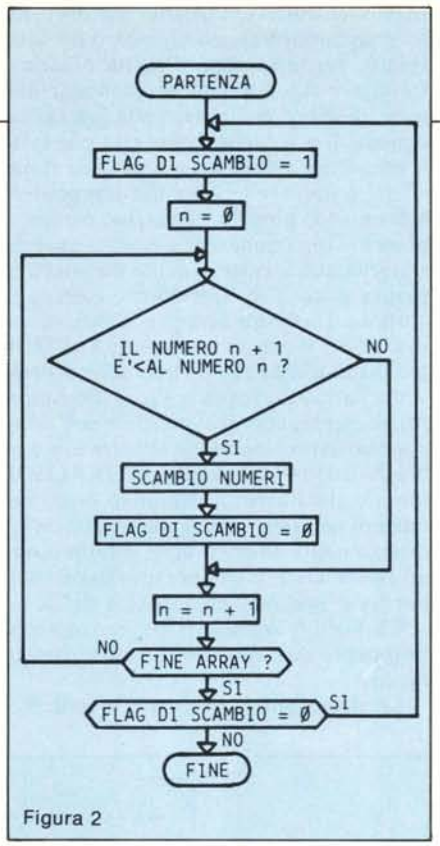

Figura 2

```
: RICERCA
   ( consente la ricerca in memoria di un numero )
   ( tra gli indirizzi ind1 ed ind2 )
   ( se n e' trovato , il suo indirizzo e' in TOS )
   ( altrimenti sara' presente 0 )
   ( ricerca positiva : ind1 ind2 n - - -  ind )
   ( ricerca negativa : ind1 ind2 n - - - 0 )
   0              ( all'inizio , poniamo in TOS 0 )
                  ( vale a dire , immaginiamo di )
                  ( non aver trovato il valore cercato )
   4 ROLL ROT     ( riordinando cosi' lo Stack )
   4 ROLL         ( 0 , ind 1 , n , ind 2 )
   3 PICK -
   2+
   0              ( impone l'inizio di DO-LOOP )
   DO
    OVER I+
    @
    OVER =
    IF
    ROT 3 PICK + I +
    SWAP ROT
   THEN
   2
   +LOOP          ( per la sequenza , vedi testo )
   DROP DROP      ( cancella i due numeri in TOS )
                  ( fine definizione )
```
Figura 1

```
: BUBBLE
  ( consente il riordino di una array di n numeri )
  ( con inizio all'indirizzo in TOS )
  ( indirizzo n - - - )
  DUP + OVER + 2 -      ( limite del DO-LOOP )
  SWAP                  ( inizio del DO-LOOP )
  >R >R                 ( conserva i limiti nel Return Stack )
                        ( vedi testo )
  BEGIN
  1
  R > R@ OVER >R        ( cerca i limiti di DO-LOOP )
  DO
   I 2 + @ I @ <        ( compara i due numeri successivi )
   IF                   ( se il 2^ numero < 1^ )
                        ( scambia i due numeri )
   I 2+ DUP @
   I DUP @
   4 ROLL ! !
   DROP 0               ( setta il flag di cambio a 0 )
  THEN
  2
 +LOOP
 UNTIL                  ( ricomincia finche' flag = 1 )
 R> R> DROP DROP        ( elimina i limiti di DO-LOOP )
                        ( fine definizione )
```
Figura 3

scente). I risultati sono stati i seguenti (in sec):

	BUBBLE	XBUBBLE
Computer APPLE IIE		
Forth su scheda	147	101
Forth su disco	159	105
Computer HEWLETT-PACKARD 87		
Forth su disco	96	68
Computer ITT 3710		
Forth su disco	125	92
Computer IBM PC		
Forth su scheda	98	71

con un vantaggio medio, quindi, del 25,30%.

È interessante anche notare come, prevedibilmente, il linguaggio su scheda sia più rapido di quello su disco e come l'HP nonostante il suo processore ad 8 bit (si tratta comunque di un sistema multiprocessore) sia altrettanto rapido dell'IBM con il suo 16 bit, dimostrando l'alta vocazione scientifica della serie 80.

Esiste però un'altra tecnica di sorting, ben più rapida, pur essendo altrettanto semplice. Si tratta dell'Insertion Sort; anche questa tecnica è abbastanza semplice, anzi più intuitiva ancora delle precedenti. In effetti è quella che usiamo comunemente quando riordiniamo un mazzo di carte. Infatti, partiamo dalla 2ª carta e la confrontiamo con la prima, eseguendo il normale bubbling, poi, passati alla successiva, ognuna di esse viene comparata con tutte le precedenti e risale nel gruppo fino al suo effettivo posto relativo prima di procedere con un altro numero. Si esegue, pertanto, proprio l'inserzione che si compie quando si riordinano le carte in mano durante una partita di canasta o di un più casereccio scopone. La figura 5 mostra la definizione di ISORT; si notino i due cicli DO-LOOP nidificati; quello più esterno scorre attraverso l'array partendo col secondo numero, e comparando ogni membro dell'array con quello precedente. Se occorre lo scambio [N(n-1) > N (n)] si passa al DO-LOOP interno che ricerca il definitivo posto del numero nell'array, lasciando inoltre un indirizzo nello stack. Inoltre il loop interno (DUP DUP 2+) apre uno spazio nell'array e, successivamente (I 3 PICK — <CMOVE !) inserisce il numero, come si fa appunto quando si riordinano le carte in mano.

Le stesse prove effettuate precedentemente sono state eseguite in tempi, con questa tecnica, ancora ridotti del 35% circa con rapporti più favorevoli per l'HP e l'IBM e con un collasso dell'Apple che, specie operando su disco, ha mostrato ridotte differenze con la tecnica precedente.

Amici Forthisti, che spero (non per mio merito, s'intende) ben più numerosi del centinaio di iscritti al FIG Italia, avete visto come è necessario avere a disposizione qualcosa di più efficiente e rapido del solito Basic o del sussiegoso Pascal. Proviamo a fare quello che abbiamo descritto in un altro linguaggio, e vediamo chi taglia il traguardo. Certo, non è facilissimo lavorarci, ma avete mai visto una corsa di formula 1 o meglio ancora una 24 ore di Le Mans? Il vincitore, alla fine della corsa, non è mica da prima pagina di Capital e non profuma certo d'acqua di Vichy, ma è arrivato primo e fa la doccia agli spettatori col magnum di champagne. Chi va in B od in P sta sotto, tra la folla, applaude, tocca con un dito le Ferrari come fossero reliquie di papa Giovanni, se tutto gli va bene rimedia un autografo di Arnoux o di Lauda, e torna a casa a vedere la domenica sportiva.

MC

```
: XSORT
  ( versione piu' efficiente di SORT )
  ( vedi testo )
  ( indirizzo n  - - - )
  DUP + OVER +    ( fine DO-LOOP = ind + 2n )
  SWAP            ( partenza DO-LOOP = indirizzo )
  >R >R           ( salva tali valori nel Return Stack )
  BEGIN
   1
   R> 2 - R@ OVER >R     ( sottrae 2 dal limite finale )
   DO
    I 2 + @ I @ <        ( compara i due numeri seguenti )
    IF
     I 2+ DUP @
                         ( se il 2^ numero < del 1^ li scambia )
     I DUP @
     4 ROLL              ( vedi testo )
     DROP 0              ( setta il flag di cambio )
    THEN
    2
   +LOOP
  UNTIL                  ( continua finche' il flag = 1 )
  R> R> DROP DROP        ( cancella i limiti di DO-LOOP )
                         ( fine definizione )
```

Figura 4

```
ISORT
( usa la tecnica dell'insertion sort )
( indirizzo  n  - - - )
DUP + OVER +          ( fine del DO-LOOP )
OVER 2+               ( inizio del DO-LOOP)
 DO
 I @ DUP              ( controlla se l'n numero e' < del precedente)
 I 2 - @ <
  IF
   0
   3 PICK 2 - I 2 -  (  limiti del loop interno = )
                     (  indirizzo - 2 ; I - 2 )
   DO
    DROP
    DUP I @ <
    IF
     I               ( se le condizioni si verificano,salva l'ind. )
    ELSE
     I 2+ LEAVE
    THEN
    -2
   +LOOP
   DUP DUP 2+        ( apre uno spazio per l'inserzione )
   I 3 PICK - <CMOVE   ( ed inserisce il numero)
  ELSE
   DROP              ( cancella il numero )
  THEN
  2
 +LOOP
DROP                 ( pulisce lo Stack )
                     ( fine definizione )
```

Figura 5

OGGI C'E'

INFO

E' UN MARCHIO INFOTEL

AL COMPLETO SERVIZIO DEI RIVENDITORI

agente esclusivo per il Lazio:

telcom

NOVITA': stampanti MITSUI 180 cps per IBM e compatibili

- *stampanti ad aghi* ***MITSUI***
- *floppy* ***MAXELL***
- *stampanti low cost* ***CP/JP - 80***
- *stampanti a margherita* ***JUKI***
- *accoppiatori acustici* ***NOVATION CAT, ANDERSON - JACOBSON*** ○ *plotter* ***YEW, ENTER C*** *digiter* ***GTCO*** ○ *mouse* ***MOUSE SYSTEM***

agente esclusivo per Lazio e Umbria:

J.soft ○ *software* ***J.soft*** *per Apple, IBM, Olivetti M24 e compatibili IBM*

GRUPPO EDITORIALE JACKSON ○ ***tutti i libri della casa editrice***

Rivenditori INFO:

A.C.S. - Roma (Ostia), via S. Canzacchi 100 - tel. 06.5627819
ALFA COMPUTER - Viterbo, via Palmanova 12c - tel. 0761.223977
BIT COMPUTERS - Roma, via Flavio Domiziano 10 - tel. 06.5126700
Roma, via F. Satolli 55/57/59 - tel. 06.6386096
Roma, viale Jonio 333/335 - tel. 06.8170632
Roma, via Nemorense 14/16 - tel. 06.858296
Roma, via Tuscolana 350/350a - tel. 06.7943980
CENTRO B - Roma, via Nomentana 332 - tel. 06.893014
COMPUMAC - Roma, viale E. Franceschini 41 - tel. 06.4563024
COMPUTIME - Roma, via Cola di Rienzo 28 - tel. 06.3581657
COSMIC - Roma, largo L. Antonelli 2 - tel. 06.5406387
DELTA COMPUTERS - Gaeta, lungomare Caboto 74 - tel. 0771.470168
FIRST SUCCESS - Latina, via A. Diaz 14 - tel. 0773.495285

INFO *- Roma, via Flavio Domiziano, 10 - Tel. 06.5126700/5138023/5127381*