Ordinamento di un vettore di variabili tipo stringa

di Carlo Iannuccelli - Velletri (Roma)

Dovendo ordinare un vettore di stringhe con un certo numero di elementi, ogni volta che si scambiano tra loro due elementi si crea della «spazzatura» con relativo problema di eliminazione e perdite di tempo (Garbage collection). La soluzione semplicissima da adottare è quella di introdurre un vettore numerico con funzione di puntatore e richiamare gli elementi del vettore stringa per mezzo di esso, scambiando i puntatori al posto delle stringhe. Non essendovi scrittura di stringhe non si ha spazzatura. È chiaro che quando si deve far riferimento ad un elemento del vettore stringa si deve sempre usare il puntatore. Ad esempio per avere l'elemento S$(x), se P(y) è il puntatore contenente x, ossia x=P(y), si deve scrivere S$(P(y)).

Il metodo di ordinamento scelto è stato quello, detto «bubble sort», delle inversioni successive, con una modifica che ritengo originale. Normalmente vi è un ciclo FOR esterno che viene ripetuto se in quello interno vi è stata una inversione. Nel ciclo FOR interno vengono sempre eseguiti tutti i confronti tra un elemento e il successivo con diminuzione di una unità ogni volta che si ripete, in quanto, ad ogni passaggio, l'elemento più grande va in cima al vettore. Il metodo realizzato consiste di un ciclo esterno, eseguito una sola volta, in cui ogni termine viene confrontato con il successivo. Quando, a seguito del confronto, si rende necessario lo scambio dell'elemento in esame con il successivo inizia un ciclo in cui quest'ultimo viene confrontato con il precedente e scambiato se necessario. Il ciclo interno termina quando l'elemento si trova al posto giusto, proseguendo con quello esterno.

È disponibile, presso la redazione, il disco con i programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 295.

```
1000 rem " ORDINAMENTO       V.1.1      25 GEN 1989
1010 :
1020 rem " ------------------------------------------------------------------
1030 rem " !
1040 rem " !       Ordinamento di un vettore mediante sottogruppi.
1050 rem " !
1060 rem " ! Variabili di competenza del programma chiamante
1070 rem " ! - V$(.) contiene gli elementi da ordinare a partire da V$(1)
1080 rem " ! - P(.) contiene i puntatori a V(.) a partire da P(1)
1090 rem " ! - WN contiene il numero degli elementi di V(.)
1100 rem " !
1110 rem " ! Variabili di competenza della routine
1120 rem " ! - W(.) serve per raggruppare due sottogruppi
1130 rem " ! - WG numero dei sottogruppi
1140 rem " ! - W1 inizio sottogruppo
1150 rem " ! - WF termine sottogruppo
1160 rem " ! - WE numero elementi per sottogruppo
1170 rem " ! - WH numero elementi per sottogruppo in fase raggruppamento
1180 rem " ! - WK numero coppie di sottogruppi ancora da raggruppare
1190 rem " ! - WS indicatore di inversione o di fine raggruppamento
1200 rem " ! - W1 W2 W3 W4 inizio e fine coppie adiacenti di sottogruppi
1210 rem " ! - G, H, I, J, K indici per i cicli
1220 rem " !
1230 rem " ! by Iannuccelli                V.1.0      15 gen 1989
1240 rem " !
1250 rem " ------------------------------------------------------------------
1260 :
1270 rem "      Calcolo numero elementi per sottogruppo e numero sottogruppi
1280 :
1290 we=wn : wg=1 : fast
1300 do
1310 : we=int((we+1)/2)
1320 : wg=wg*2
1330 loop until we<50
1340 :
1350 rem "      Ordinamento all'interno dei sottogruppi
1360 :
1370 for g=1 to wg
1380 :    w1=1+we*(g-1) : wf=we*g
1390 :    if wf>wn then wf=wn
1400 :    if wf<>w1 then begin
1410 :       for h=w1 to wf-1
1420 :           if v$(p(h))>v$(p(h+1)) then begin
1430 :              p(0)=p(h) : p(h)=p(h+1) : p(h+1)=p(0)
1440 :              k=h
1450 :              do while k>w1
1460 :                 if v$(p(k-1))>v$(p(k)) then begin
1470 :                    p(0)=p(k-1): p(k-1)=p(k): p(k)=p(0)
1480 :                    k=k-1
1490 :                 bend: else exit
1500 :              loop
1510 :           bend
1520 :       next
1530 :    bend
1540 next
1550 :
1560 rem "      Raggruppamento finale
1570 :
1580 wk=wg : wh=we
1590 do
1600 : w4=0
1610 : wk=wk/2
1620 : for g=1 to wk
1630 :     w1=w4+1
1640 :     w3=w1+wh
1650 :     w2=w3-1
1660 :     w4=w2+wh
1670 :     if w4>wn then w4=wn
1680 :     w1=w1 : wf=w4
1690 :     i=w1 : j=w3 : k=w1 : ws=0
1700 :     do
1710 :        if i>w2 then begin
1720 :           do while j<=w4
1730 :              w(k)=p(j) : j=j+1 : k=k+1
1740 :           loop
1750 :           ws=1
1760 :        bend
1770 :        if j>w4 then begin
1780 :           do while i<=w2
1790 :              w(k)=p(i) : i=i+1 : k=k+1
1800 :           loop
1810 :           ws=1
1820 :        bend
1830 :        if ws=1 then exit
1840 :        if v$(p(i))>v$(p(j)) then w(k)=p(j): j=j+1: else w(k)=p(i): i=i+1
1850 :        k=k+1
1860 :     loop
1870 :     for l=w1 to wf
1880 :        p(l)=w(l)
1890 :     next
1900 :  next
1910 :  wh=wh*2
1920 loop until wk=1
1930 :
1940 return
1950 :
1960 rem " Fine routine Ordinamento
1970 :
```

Routine ordinamento vettori.

```
10 rem "Programma di prova ordinamento  28/1/89
15 :
16 input "quanti";wn
20 dim v$(wn),p(wn), w(wn)
30 for i=1 to wn
40 :   p(i)=i
50 :   v$(wn+1-i)=str$(99+i)
60 next
70 ti$="000000"
80 gosub 1270
90 scnclr
100 print ti$
110 for i=1 to wn
120 : print v$(p(i));
130 next
140 slow:end
1000 rem " VETTORE.SORT        V.1.1      25 GEN 1989
1010 :
1020 rem " ------------------------------------------------------------------------
1030 rem " |                                                                      |
1040 rem " |       Ordinamento di un vettore mediante sottogruppi.                |
1050 rem " |                                                                      |
1060 rem " |  Variabili di competenza del programma chiamante                     |
1070 rem " |  - V$(.) contiene gli elementi da ordinare a partire da V$(1)        |
1080 rem " |  - P(.) contiene i puntatori a V(.) a partire da P(1)                |
1090 rem " |  - WN contiene il numero degli elementi di V(.)                      |
1100 rem " |                                                                      |
1110 rem " |  Variabili di competenza della routine                               |
1120 rem " |  - W(.) serve per raggruppare due sottogruppi                        |
1130 rem " |  - WG numero dei sottogruppi                                         |
1140 rem " |  - WI inizio sottogruppo                                             |
1150 rem " |  - WF termine sottogruppo                                            |
1160 rem " |  - WE numero elementi per sottogruppo                                |
1170 rem " |  - WH numero elementi per sottogruppo in fase raggruppamento         |
1180 rem " |  - WK numero coppie di sottogruppi ancora da raggruppare             |
1190 rem " |  - WS indicatore di inversione o di fine raggruppamento              |
1200 rem " |  - W1 W2 W3 W4 inizio e fine coppie adiacenti di sottogruppi         |
1210 rem " |  - G, H, I, J, K indici per i cicli                                  |
1220 rem " |                                                                      |
1230 rem " |  by Iannuccelli                V.1.0       15 gen 1989               |
1240 rem " |                                                                      |
1250 rem " ------------------------------------------------------------------------
1260 :
1270 rem "     Calcolo numero elementi per sottogruppo e numero sottogruppi
1280 :
1290 we=wn : wg=1 : fast
1300 do
1310 : we=int((we+1)/2)
1320 : wg=wg*2
1330 loop until we<50
1340 :
1350 rem "     Ordinamento all'interno dei sottogruppi
1360 :
1370 for g=1 to wg
1380 :   wi=1+we*(g-1) : wf=we*g
1390 :   if wf>wn then wf=wn
1400 :   if wf<>wi then begin
1410 :      for h=wi to wf-1
1420 :          if v$(p(h))>v$(p(h+1)) then begin
1430 :             p(0)=p(h) : p(h)=p(h+1) : p(h+1)=p(0)
1440 :             k=h
1450 :             do while k>wi
1460 :                if v$(p(k-1))>v$(p(k)) then begin
1470 :                   p(0)=p(k-1): p(k-1)=p(k): p(k)=p(0)
1480 :                   k=k-1
1490 :                bend: else exit
1500 :             loop
1510 :          bend
1520 :      next
1530 :   bend
1540 next
1550 :
1560 rem "      Raggruppamento finale
1570 :
1580 wk=wg : wh=we
1590 do
1600 : w4=0
1610 : wk=wk/2
1620 : for g=1 to wk
1630 :      w1=w4+1
1640 :      w3=w1+wh
1650 :      w2=w3-1
1660 :      w4=w2+wh
1670 :      if w4>wn then w4=wn
1680 :      wi=w1 : wf=w4
1690 :      i=w1 : j=w3 : k=wi : ws=0
1700 :      do
1710 :         if i>w2 then begin
1720 :            do while j<=w4
1730 :               w(k)=p(j) : j=j+1 : k=k+1
1740 :            loop
1750 :            ws=1
1760 :         bend
1770 :         if j>w4 then begin
1780 :            do while i<=w2
1790 :               w(k)=p(i) : i=i+1 : k=k+1
1800 :            loop
1810 :            ws=1
1820 :         bend
1830 :         if ws=1 then exit
1840 :         if v$(p(i))>v$(p(j)) then w(k)=p(j): j=j+1: else w(k)=p(i): i=i+1
1850 :         k=k+1
1860 :      loop
1870 :      for l=wi to wf
1880 :          p(l)=w(l)
1890 :      next
1900 :   next
1910 :    wh=wh*2
1920 loop until wk=1
1930 :
1940 return
1950 :
1960 rem " Fine routine Ordinamento
1970 :slow

ready.
```

Programma di prova per ordinamento vettori.

```
100 rem prova ordinamento nuovo programma 27/5/89
110 scnclr
120 input"quanti";n
130 fast
140 dim a$(n),p(n)
150 for i=1to n
160 :  p(i)=i : a$(n+1-i)=str$(99+i)
170 next
180 ti$="000000"
190 for i=1 to n -1
200 :   if a$(p(i))>a$(p(i+1)) then begin
210 :      j=i : p(0)=p(i):p(i)=p(i+1) :p(i+1)=p(0)
220 :      do while j>1
230 :        if a$(p(j-1))>a$(p(j)) then p(0)=p(j):p(j)=p(j-1):p(j-1)=p(0):j=j-1
    :        else exit
240 :      loop
250 :   bend
260 next
270 print"(home)"
280 print ti$
290 for m=1 to n
300 :   print a$(p(m));
310 next
320 slow

ready.
```

Prova ordinamento vettori nuovo programma.

Con tale metodo il numero delle volte che viene eseguita la routine è uguale a quello del bubble sort solo nel caso in cui una lista già ordinata deve essere invertita, altrimenti è sempre inferiore. Tale numero massimo è n(n+1)/2.

Malgrado questo accorgimento, dovendo ordinare molti elementi i tempi crescono in modo esponenziale. Ho provato con una serie da invertire completamente, quindi con tempi massimi, e ho ottenuto per 100 elementi un tempo di 2 minuti e 6 secondi, per 200 elementi un tempo di 8 minuti e 25 secondi, per 300 elementi 18 minuti e 38 secondi...

Ho studiato quindi una routine in grado di rendere i tempi di ordinamento accettabili. L'idea è stata semplice: suddividere il vettore dei puntatori in più parti, ordinarli separatamente e poi fonderli. La realizzazione (poche istruzioni che per essere più chiare sono state scritte una per riga e con molti spazi: basterà compattarla per diminuire i tempi di elaborazione) consiste nella divisione del vettore dei puntatori in segmenti aventi meno di 50 elementi. Il numero dei segmenti viene ottenuto dividendo successivamente il numero degli elementi da ordinare per 2 (istruzioni 1270-1330): tale numero è una potenza di 2. Quindi nella determinazione dell'inizio e fine di ogni segmento e al relativo ordinamento (istruzioni 1350 - 1550). Infine nel procedere al raggruppamento finale (istruzioni 1560-1930), tale raggruppamento viene ottenuto fondendo, su un vettore di transito, il primo segmento con il secondo, il terzo con il quarto, e così di seguito, ottenendo un numero di segmenti che è la metà di quello iniziale. Si trasferisce il vettore di transito su quello dei puntatori e si ripete il ciclo fino ad avere un unico segmento. Con tale routine il tempo di ordinamento degli elementi considerati prima è stato: per 100 elementi di 42 secondi, per 200 elementi di 1 minuto e 30 secondi, per 300 elementi di 3 minuti e 9 secondi. Provare per credere!

La routine scritta in modo strutturato può essere trascritta sia in Pascal che in altri Basic. Per il GWBASIC occorrerà inserire i gruppi IF ... BEGIN ... BEND in una unica riga facendo attenzione all'accoppiamento con ELSE, modificare i cicli nell'unico modo WHILE e trovare come sostituire l'EXIT: in altre parole riscriverla!

```
10 rem "PROVA INPUT      Iannuccelli    V.1.0   28 maggio 1989
20 :
30 iz$(0)="Allineamento a sinistra   IZ=0"
40 iz$(1)="Allineamento a destra     IZ=1"
50 ic$(0)="Nessuna modifica          IC=0"
60 ic$(1)="Prima carattere maiuscolo IC=1"
70 ic$(2)="Tutto maiuscolo           IC=2"
80 rem graphic 5,1
90 rem color 7,12: color 7,12
100 :
110 print chr$(14)
120 do
130 :  scnclr
140 :  char 1,0,0,"Colonna   IX "
150 :  input ix
160 :  char 1,0,1,"Riga      IY "
170 :  input iy
180 :  char 1,0,2,"Lunghezza IL "
190 :  input il
200 :  for iz=0 to 1
210 :    char 1,0,4,iz$(iz)
220 :    for ic=0 to 2
230 :      char 1,0,5,ic$(ic)
240 :      gosub 600
250 :      char 1,0,7,"Premere un tasto"
260 :      getkey a$
270 :      char 1,0,7,"                 "
280 :    next
290 :  next
300 loop
310 :
320 rem " *   INPUT STRINGA            V.2.1      12 gen 1989  *
330 :
340 rem " ____________________________________________________________
350 rem " |                                                          |
360 rem " | INPUT STRINGA DI LUNGHEZZA E POSIZIONE ASSEGNATA         |
370 rem " |                                                          |
380 rem " | Parametri da fornire: IX   Colonna di inizio             |
390 rem " |                       IY   Riga di inizio                |
400 rem " |                       IL   Lunghezza della stringa       |
410 rem" |                       IZ   Allineamento                  |
420 rem" |                            0 : a sinistra                |
430 rem" |                            1 : a destra                  |
440 rem" |                       IC   Modifica                      |
450 rem" |                            0 : nessuna                   |
460 rem" |                            1 : primo caratt. maiuscolo   |
470 rem" |                            2 : tutto maiuscolo           |
480 rem " |                                                         |
490 rem " | Variabile al rientro: IS$  Stringa lunga IL             |
500 rem " |                                                         |
510 rem " |                                                         |
520 rem " | Tasti speciali accettati: RETURN e DEL                  |
530 rem " |                                                         |
540 rem " | Variabili usate:  UA   UD   UL   UA$   UB$              |
550 rem " |                                                         |
560 rem " |                              by Carlo Iannuccelli 1988  |
570 rem " ____________________________________________________________
580 :
590 :
600 ul=0:is$="":ie$=""
610 print chr$(14)
620 ub$="
                                                                    "
630 is$=left$(ub$,il)
640 poke 208,0
650 do
660 :  do
670 :    char 1,ix,iy,is$,1
680 :    ud=1
690 :    getkey ua$
700 :    ua=asc(ua$)
710 :    if ua=20 then ua=128: ud=-1
720 :    if ua=34 then ua=0
730 :    if (ua>31 and ua<129) or (ua>160 and ua<255) then
         begin
740 :      if ua>64 and ua<91 then
           begin
750 :        if ic=1 and ul=0  then ua$=chr$(ua+32)
760 :        if ic=2 then ua$=chr$(ua+32)
           bend
780 :      ul=ul+ud
790 :      if ul<0 then ul=0
800 :      if ul>il then ul=il
810 :      ie$=ie$+ua$ : ie$=left$(ie$,ul)
820 :      if iz=0 then is$=left$(ie$+ub$,il) :else is$=right$(ub$+ie$,il)
830 :    bend: else exit
840 :  loop
850 loop until ua$=chr$(13) and ul>0
860 return

ready.
```

Programma di prova per routine input stringa.

Input di una stringa

di Carlo Iannuccelli - Velletri (RM)

Spesse volte mi sono trovato nella necessità di far apparire la richiesta di un dato in uno specifico punto del monitor, evidenziandone la lunghezza massima e se veniva scritto da sinistra verso destra o da destra verso sinistra.

La routine che presento provvede a tale scopo; la lunghezza viene segnalata dalla zona in reverse, ma potrebbe essere sostituita da una serie di trattini o altro.

La stringa viene restituita con la lunghezza richiesta e questo per poter preparare dei record da stampare direttamente secondo un tracciato preparato.

```
1000 rem " *   INPUT STRINGA            V.2.1      12 gen 1989  *
1010 :
1020 rem " ____________________________________________________________
1030 rem " |                                                           |
1040 rem " | INPUT STRINGA DI LUNGHEZZA E POSIZIONE ASSEGNATA          |
1050 rem " |                                                           |
1060 rem " | Parametri da fornire: IX   Colonna di inizio              |
1070 rem " |                       IY   Riga di inizio                 |
1080 rem " |                       IL   Lunghezza della stringa        |
1090 rem" |                       IZ   Allineamento                   |
1100 rem" |                            0 : a sinistra                 |
1110 rem" |                            1 : a destra                   |
1120 rem" |                       IC   Modifica                       |
1130 rem" |                            0 : nessuna                    |
1140 rem" |                            1 : primo caratt. maiuscolo    |
1150 rem" |                            2 : tutto maiuscolo            |
1160 rem " |                                                          |
1170 rem " | Variabile al rientro: IS$  Stringa lunga IL               |
1180 rem " |                                                           |
1190 rem " |                                                           |
1200 rem " | Tasti speciali accettati: RETURN e DEL                    |
1210 rem " |                                                           |
1220 rem " | Variabili usate:  UA   UD   UL   UA$   UB$                |
1230 rem " |                                                           |
1240 rem " |                              by Carlo Iannuccelli 1988    |
1250 rem " |___________________________________________________________|
1260 :
1270 :
1280 ul=0:is$="":ie$=""
1290 print chr$(14)
1300 ub$="                                                              "

1310 is$=left$(ub$,il)
1320 poke 208,0
1330 do
1340 :  do
1350 :     char 1,ix,iy,is$,1
1360 :     ud=1
1370 :     getkey ua$
1380 :     ua=asc(ua$)
1390 :     if ua=20 then ua=128: ud=-1
1400 :     if ua=34 then ua=0
1410 :     if (ua>31 and ua<129) or (ua>160 and ua<255) then
           begin
1420 :       if ua>64 and ua<91 then
             begin
1430 :          if ic=1 and ul=0  then ua$=chr$(ua+32)
1440 :          if ic=2 then ua$=chr$(ua+32)
1450 :       bend
1460 :       ul=ul+ud
1470 :       if ul<0 then ul=0
1480 :       if ul>il then ul=il
1490 :       ie$=ie$+ua$ : ie$=left$(ie$,ul)
1500 :       if iz=0 then is$=left$(ie$+ub$,il) :else is$=right$(ub$+ie$,il)
1510 :     bend: else exit
1520 :  loop
1530 loop until ua$=chr$(13) and ul>0
1540 return

ready.
```

Routine input stringa.

Una nota sul programma Sogar 128

Vi ricordate il programma Sogar pubblicato sul numero 82 di MC? Bene! È stato scoperto che su alcuni computer tale programma non funziona. L'anomalia è dovuta ai differenti tipi di tastiera che si trovano in giro; l'anomalia (per la precisione il blocco del sistema!) si verifica su computer che non hanno la tastiera QZERTY. In pratica c'è qualche differenza nella gestione delle routine di tastiera da parte del Kernal e, dato che Sogar utilizza proprio queste routine, succede un patatrack!

Per riportare tutto alla normalità, digitate il programmino che segue.
Prima di dare il run inserite il dischetto contenente il Sogar nel drive (consigliamo per sicurezza di fare prima una copia del dischetto).

```
10 BANK 1
20 BLOAD"SOGAR/2",B1:POKE 7896,197:
   SCRATCH"SOGAR/2":BSAVE"SOGAR/2",B1,P4864 TO P14336
30 BLOAD"SOGAR/C",B1:POKE 7896,197:
   SCRATCH"SOGAR/C":BSAVE"SOGAR/C",B1,P4864 TO P14336
```