Sorting Code

The Shell Sort is more efficient than either the Bubble or Insertion Sorts for long arrays. It works by dividing the data into a series of 'chains'

On page 286 we looked at two methods of sorting an array into order — the Bubble and Insertion Sorts. Generally, the Bubble Sort is easier to carry out, but the Insertion Sort is faster. Experience of these two methods shows that what takes the time is swapping cards around over short distances: it's usually far better to swap once over a long distance than several times over short distances.

```
7999 REM**********************
8000 REM*        SHELL        *
8001 REM**********************
8025 PRINT "SHELL SORT - GO !!!!!"
8050 LET LK=LT
8100 FOR Z=0 TO 1 STEP 0
8150 LET LK=INT(LK/II)
8200 FOR LB=1 TO LK
8250 LET LL=LB+LK
8300 FOR P=LL TO LT STEP LK
8350 LET D=R(P)
8400 FOR Q=P TO LL STEP-LK
8450 LET R(Q)=R(Q-LK)
8500 IF D<=R(Q) THEN LET R(Q)=D:LET Q=LL
8550 NEXT Q
8600 IF D>R(LB) THEN LET R(LB)=D
8650 NEXT P
8700 NEXT LB
8750 IF LK=1 THEN LET Z=1
8800 NEXT Z
8850 PRINT "SHELL SORT - STOP !!!!!"
8900 RETURN
```

To add this routine to the sorting demonstration program on page 287, change line 350 to:

```
350 LET I=1:LET O=0:LET II=+:LET TH=3
```

and change line 900 to:

```
900 ON SR GOSUB 6000,7000,8000
```

A better method than either of these two is called the 'Shell Sort' (named after its creator, D Shell). This method ensures that the disorder in the array is reduced early in the sort (so that items are not a long way from their true positions), and enables swaps to operate over relatively long distances. Here is a method for this sort:

1) Lay out all the cards of one suit in any order. They are to be sorted into descending order so that the King will be the leftmost card, and the Ace the rightmost. Count the cards, divide that number (in this case, 13) by two, ignoring any remainder, and write the result (i.e. six) on a piece of paper labelled 'The Link'.

2) Place a fivepence piece under the leftmost card (call this Position One), and a tenpence piece in the Link position (i.e. Position Six in the first instance). All of the cards from the First to the Link position are each to be the leftmost end cards in a series of 'chains' of cards. The number of chains will equal the current value of the Link. Each chain is formed by starting with its end card, adding the Link to the end card's position number to get the position of the next card, adding the Link to get the position of the next card, and so on until the end of the array has been reached or exceeded. The first chain, therefore, comprises the cards in positions One, Seven, and Thirteen; the second chain is the cards in positions Two and Eight; the third is the cards in positions Three and Nine. The last chain is the cards in positions Six (the present value of the Link) and Twelve.

3) Now, having marked the boundaries with the five- and tenpence pieces, push the cards that comprise the first chain out of the array so that you can see them in isolation, and sort them into order using either the Bubble or Insertion Sort as described on page 286 (the listing with this article uses the Insertion method).

4) Push the ordered chain back into the gaps in the array, and repeat the above with the next chain, and the next, and so on, until all the chains whose leftmost cards lie between the five- and ten-pence pieces have been sorted.

5) When all the chains have been sorted, divide the Link by two, ignoring any remainder. If the Link is now less than one then the array will be sorted. Otherwise, repeat from Step Two above with the new value of the Link.

Shell Sort Panel

Position No.									Link Value	Comments
1	2	3	4	5	6	7	8	9		
2	8	9	3	T	5	K	6	7	(9/2)=>4	Begin Pass
*	+	@	$	*	+	@	$	*		Form chains
T			7					2		Sort Chain 1
	8			5						Sort Chain 2
		K			9					Sort Chain 3
			6			3				Sort Chain 4
T	8	K	6	7	5	9	3	2		Begin Pass
T	8	K	6	7	5	9	3	2	(4/2)=>2	End of Pass
*	+	*	+	*	+	*	+	*		Form Chains
K		T		9		7		2		Sort Chain 1
	8		6		5		3			Sort Chain 2
K	8	T	6	9	5	7	3	2		End of Pass
K	8	T	6	9	5	7	3	2	(2/2)=>1	Begin Pass
*	*	*	*	*	*	*	*	*		Form Chain 1
K	T	9	8	7	6	5	3	2		End of Pass

KEY	
*	Member of Chain 1
+	Member of Chain 2
@	Member of Chain 3
$	Member of Chain 4

Shell Sort

The example of the Shell Sort for a reduced hand that we show in the panel demonstrates its unique method of dividing the array into a series of chains (with spacings based on the current Link number). These chains are separately sorted, in this case using the Insertion method, before a pass is completed.

The program listing for a Shell Sort given here must be used in conjunction with the testbed program on page 287. When we tested it, there was a significant improvement over the other sorting methods once the number of items to be sorted exceeded 40