How to perform a Shell Sort

Shell Sort is really just an extension of Insertion Sort, with two observations in mind:

- Insertion Sort is efficient if the input is "almost sorted".
- Insertion Sort is inefficient, on average, because it moves values just one position at a time.

Shell Sort is similar to Insertion Sort, but it works by taking "bigger steps" as it rearranges values, gradually decreasing the step size down towards one. In the end, Shell Sort performs a regular Insertion Sort but by then, the array of data is guaranteed to be "almost sorted".

Consider a small value that is initially stored in the wrong end of the array. Using Insertion Sort, it will take roughly N comparisons and exchanges to move this value all the way to the other end of the array. With Shell Sort, we'll first move values using giant step sizes, so a small value will move a long way towards it's final position, with just a few comparisons and exchanges.

```
procedure SortShell(var aSort: array of Word);
var
  iI, iJ, iK,
  iSize: Integer;
  wTemp: Word;
begin
  iSize := High(aSort);
  iK := iSize shr 1;
  while iK > 0 do begin
    for iI := 0 to iSize - iK do begin
      iJ := iI;
      while (iJ >= 0) and (aSort[iJ] > aSort[iJ + iK]) do begin
        wTemp := aSort[iJ];
        aSort[iJ] := aSort[iJ + iK];
        aSort[iJ + iK] := wTemp;
        if iJ > iK then
          Dec(iJ, iK)
        else
          iJ := 0
      end;
    end;
    iK := iK shr 1;
  end;
end;
```

Author:	Unknown
Added:	2007-06-02
Last updated:	2010-03-16