ShellSort

ShellSort ist ein von **Donald L. Shell** im Jahre 1959 entwickeltes Sortierverfahren, das auf dem Sortierverfahren des direkten Einfügens (InsertionSort, Sortieren durch Einfügen) basiert.

ShellSort bedient sich prinzipiell **Insertionsort**. Es versucht den Nachteil auszugleichen, dass hier Elemente in der Sequenz oft über weite Strecken verschoben werden müssen. Dies macht InsertionSort ineffizient.

ShellSort verfolgt den Ansatz, dass die Sequenz z.B. erst 4-sortiert wird, dann 2-sortiert, und zuletzt mit normalem InsertionSort sozusagen 1-sortiert. Man spricht von **h-sortierten** Daten.

Anschaulich wäre dies anhand von Hilfsmatrizen darzustellen (siehe Beispiel):
- Die Daten werden in eine k-spaltige Matrix geschrieben
- Die Spalten der Matrix werden einzeln sortiert

Daraus resultiert eine grobe Sortierung. Dieser Schritt wird **mehrmals wiederholt**, wobei jeweils die **Breite der Matrix verringert** wird, bis die Matrix nur noch aus einer einzigen vollständig sortierten Spalte besteht.

ShellSort arbeitet **in-place**, gehört jedoch nicht zu den stabilen Sortieralgorithmen.

Zu sortieren sind die Zahlen "**2 5 3 4 3 9 3 2 5 4 1 3**" mittels der Folge **2n, ..., 4, 2, 1**.

Beispiel:
Zuerst werden die Daten in eine Matrix mit 4 Spalten eingetragen und spaltenweise sortiert. Die Zahlenfolge wird also 4-sortiert.

```
2 5 3 4      2 4 1 2
3 9 3 2  ->  3 5 3 3
5 4 1 3      5 9 3 4
```

Die sortierte 4-Spalten-Matrix wird nun in zwei Spalten aufgeteilt. Diese Spalten werden nun 2-sortiert.

```
2 1      2 1
3 3      3 2
5 3  ->  4 3
4 2      5 3
5 3      5 3
9 4      9 4
```

Die sortierte 2-Spalten-Matrix wird nun in eine Spalte geschrieben und wieder sortiert mittels normalem InsertionSort. Der Vorteil dabei besteht darin, dass kein Element der Sequenz so weit verschoben werden muss, wie bei InsertionSort, das auf eine völlig unsortierte Folge losgelassen wird.

2 3 4 5 5 9 1 2 3 3 3 4 -> 1 2 2 3 3 3 3 4 4 5 5 9

Die verwendete Schrittfolge **1, 2, 4, 8, 16, ..., 2n** (wie original 1959 von Shellvorgeschlagen) erweist sich in der Praxis als **nicht zweckmäßig**, da nur gerade Stellen sortiert werden und ungerade Stellen der Sequenz nur im letzten Schritt angefasst werden.

Als **zweckmäßiger** hat sich **1, 4, 13, ..., 3*(n - 1) + 1** erwiesen.
Ein Array wird sortiert und vorher und nachher ausgegeben.

```
#include <stdio.h>
inline void tausche(int *a, int *b) {
   int c = *a;
   *a = *b;
   *b = c;
}

void sort(int array[], size_t size) {
   size_t step=size -1;
   size_t start,i,j;
   do {

      step/=2;
      for(start=0; start<step; ++start){

         for(i=start+1; i<size; i+=step){

            for(j=i-1; j>=0; j-=step){

               if((j+step)<size && array[j]>array[j+step])
                  tausche(&array[j],&array[j+step]);
               else
                  break;
            }
         }
      }
   } while(step>0);
}

int main(void){

  int n = 0;
  int x[] = {30,5,24,11,26,20,4,23,9,25,6,28,15,27,7,22,10,3,1,13,21,29,
             17,2,19,8,16,14,12,18};
  printf("Shellsort...\nVorher: ");
  for (n = 0; n < sizeof(x)/sizeof(int); n++) printf("%d|",x[n]);

  sort(x,sizeof(x)/sizeof(int));

  printf("\n\nNachher:");
  for (n = 0; n < sizeof(x)/sizeof(int); n++) printf("%d|",x[n]);
  getc(stdin);
  return 0;
}
```

```
Shellsort...
Vorher: 30|5|24|11|26|20|4|23|9|25|6|28|15|27|7|22|10|3|1|13|21|29|17|2|19|8|16|14|12|18|

Nachher:1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|
```

```c
#include  <stdio.h>
#include <stdlib.h>
#define N 100

void shell_sort(int z[], int l, int r) {
   int i, j, k, h, v;
   int sw[16] = { 1391376, 463792, 198768, 86961, 33936, 13776,
                  4592, 1968, 861, 336, 112, 48, 21, 7, 3, 1 };
   for (k=0; k<16; k++) {
      for (h=sw[k], i=l+h; i<=r; i++) {
         v = z[i];
         j = i;
         while (j >= h && z[j-h] > v) {
            z[j] = z[j-h];
            j -= h;
         }
         z[j] = v;
      }
   }
}

int  main(int argc, char *argv[]) {
   int i, x, zahlen[N], h[N+1]={0};
   for (i=0; i<N; i++) {
      do { x=rand()%N+1; } while (h[x]);
      zahlen[i] = x;
      h[x] = 1;
   }
   printf("ShellSort\n\nunsortiert:\n");
   for(i=0; i<N; i++) printf("%d ", zahlen[i]);

   shell_sort(zahlen, 0, N-1);

   printf("\n\nsortiert:\n");
   for(i=0; i<N; i++) printf("%d ", zahlen[i]);
   getc(stdin);
   return 0;
}

/*
ShellSort

unsortiert:
42 68 35 1 70 25 79 59 63 65 6 46 82 28 62 92 96 43 37 5 3 54 93 83 22 17 19 48
27 72 39 13 100 36 95 4 12 23 34 74 69 45 58 38 60 24 30 91 89 7 41 49 47 71 51
2 94 85 55 57 67 77 32 9 40 16 31 78 87 73 98 56 75 53 8 88 84 10 14 11 21 97 29
18 15 52 50 20 99 90 86 44 81 80 76 33 26 61 64 66

sortiert:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
*/
```