

## Sortieren

Es ist ein Programmsystem zu schreiben, mit dessen Hilfe man die folgenden Sortierverfahren hinsichtlich ihrer Effektivität untersuchen kann:

### 1. Sortieren mittels

Einfügen (Insertion-Sort),  
 Auswählen (Selection-Sort)  
 Austauschen (Bubble-Sort, Exchange-Sort) ohne Merker  
 Austauschen (Bubble-Sort, Exchange-Sort) mit Merkern

### 2. Sortieren mittels

abnehmender Schrittweite (Shellsort),  
 Bäumen (Heapsort),  
 Zerlegung (Quicksort),  
 Mischen (Mergesort).

Als Parameter sind zu wählen :

- a. sortierte Schlüssel
- b. zufällig angeordnete Schlüssel
- c. umgekehrt sortierte Schlüssel

Wählen Sie die Anzahl der Schlüssel  $n$  geeignet hinsichtlich der Schnelligkeit Ihres Rechners und der Verarbeitungsdauer. Testen Sie die Verfahren auch für  $2n$ ,  $3n$  und  $4n$  Schlüssel. Es muss möglich sein, für die Verfahren der Gruppe 1 und 2 separate  $n$  eingeben zu können.

Im Falle b) sind die Schlüssel quasizufällige Zahlen (generiert mit einem Zufallszahlengenerator). Diese Zufallszahlen sind **vor dem Testen** zu sortieren und in den Fällen a) und c) entsprechend zu verwenden. Alle Verfahren der Gruppe 1 und der Gruppe 2 müssen die **gleichen** Zufallszahlen zugrunde liegen (d.h. für jede Gruppe nur **einmal** generieren)

Gestalten Sie die Ausgabe tabellarisch in 4 Tabellen (für  $n$ ,  $2n$ ,  $3n$  und  $4n$ ) wie folgt:

No. of keys = ...	ascending [ms]	random [ms]	descending [ms]
-----			
Insertion-Sort	x	x	x
Selection-Sort	x	x	x
Bubble-Sort	x	x	x
Bubble-Sort w/flags	x	x	x
No. of keys = ...			
Shell-Sort	x	x	x
Heap-Sort	x	x	x
Quick-Sort	x	x	x
Merge-Sort	x	x	x

## Empfehlungen:

1. Arbeiten Sie mit dynamischen Arrays
2. Bei Mergesort: Brauchen Sie ein Hilfsfeld, generieren Sie dieses nur einmal, nicht bei jeder Rekursion neu
3. Vorgehensweise bei der Programmierung der Zeitmessung:

a) Java:

```
import java.util.*;

long TA=System.currentTimeMillis();    // Anfangszeit
...
//Aufruf der Sortierfunktion
...
long TE=System.currentTimeMillis();    // Endzeit
long DE=TE-TA;
```

b) C++:

```
//CTimMess.h//
#include <sys/timeb.h>
#include <time.h>

#define MillTime struct _timeb    //Millisekunden
typedef time_t SekTime;          //Sekunden
class CTimMess {
private:
    //MilliTime
    MillTime    BegmTime, EndmTime; //gesamt Zeit seit 1970 in Sekunden
    double DiffTime;
    SekTime BegsTime, EndsTime;
public:
    void BeginMessung();
    void EndeMessung();
    double Differenz();
};

//CTimMess.cpp//
#include "CTimMess.h"
#define MillTime _ftime64_s

void CTimMess :: BeginMessung() {
    MillTime (&BegmTime);
    time (&BegsTime);
}

void CTimMess :: EndeMessung() {
    MillTime (&EndmTime);
    time (&EndsTime);
}

double CTimMess :: Differenz() {
    return (double)1000*(EndsTime-BegsTime)+EndmTime.millitm -
           BegmTime.millitm ;
}
```