2. Übungsblatt

4. Aufgabe:

Implementieren Sie REKURSIVE-Varianten der Bubble-Sort und Quicksort-Algorithmen. Achten Sie neben der Korrektheit auch auf die Effizienz (Geschwindigkeit, Vermeidung überflüssiger Anweisungen) des Programms.

Führen Sie in Ihrem Programm Zeitmessungen durch. Gesucht wird wie lange es dauert,

- die zu sortierenden Zahlen einzulesen,
- die Zahlen zu sortieren und
- die Zahlen sortiert in einen Datei zu schreiben sowie
- das gesamte Programm auszuführen.

Zur Zeitmessung sollen Sie folgende Prozeduren verwenden:

```
#include <dos.h>
void gettime(struct time *timep);
void settime(struct time *timep);

/*  Lesen und Setzen der Systemzeit.
    gettime liefert in der über timep angegebenen
    Struktur des Typs time die aktuelle Uhrzeit des
    Systems zurück.

    settime setzt die Uhrzeit des Systems auf die
    Werte, die in der über timep angegebenen Struktur
    des Typs time enthalten sind. */
```

Die Struktur **`struct time`** ist folgendermaßen definiert:

```
struct time {
   unsigned char ti_min;     /* Minuten */
   unsigned char ti_hour;    /* Stunden */
   unsigned char ti_hund;    /* 1/100 Sekunden */
   unsigned char ti_sec;     /* Sekunden */
};
```

Damit die Ergebnisse der verschiedenen Implementierungen verglichen werden können, sollen Sie die zu sortierenden Zahlen in einer Datei vorgeben. Die Datei

soll zunächst eine Zahl enthalten, die angibt, wie viele Zahlen noch folgen bzw. sortiert werden sollen und dann alle diese Zahlen. Sie sollten davon ausgehen, daß bis zu 10 000 Zahlen sortiert werden sollen.

5. Aufgabe:

Nach dem Sie in der vorigen Aufgabe die Zahlen sortiert haben, ist nun ein Programm zu schreiben, daß feststellt, ob eine bestimmte Zahl in dieser Menge vorkommt. Es soll eine beliebige Zahl eingegeben werden können. Ist diese Zahl vorhanden, dann wird der Index zu der Zahl in dem Feld, in dem die Zahlen gehalten werden, ausgegeben, sonst wird eine Meldung ausgegeben, daß die Zahl nicht vorkommt.

Achten Sie wieder darauf, daß der Algorithmus effizient arbeitet. Benutzen Sie deshalb eine binäre Suche, die Sie:

a) iterativ
b) rekursiv

durchführen.

6. Aufgabe (Praktikum)

Implementieren Sie REKURSIVE-Varianten des Quicksort- und Merge-Sort-Algorithmus. Die Daten sollen hierbei im Gegensatz zu Aufgabe 4 nicht in einem Feld sondern in einer verketteten Liste gehalten werden. Hierzu sollen zuerst alle Daten einer Datei (z.B. Daten.txt) in eine Liste einlesen werden. Diese Liste wird dann entweder mittels Quicksort oder Merge-Sort sortiert und dann wieder in einer Datei (z.B. Daten.quick bzw. Daten.merge) abgelegt. Messen Sie auch bei dieser Aufgabe die Zeit, die der Sortiervorgang braucht!

Hinweis: Da Ihr Programm intern kein Feld benutzt sollte es auch mit sehr großen Datenmengen (>> 10.000) zurechtkommen. Überprüfen Sie dies!