

## Übung 8

### Quicksort

Nadine Schärmann & Thomas Klütz

”Ein **stabiles Sortierverfahren** ist ein Sortieralgorithmus, der die Reihenfolge der Datensätze, deren Sortierschlüssel gleich sind, bewahrt.”

Zum Beispiel:

- Bubblesort, Insertionsort, Mergesort
- nicht stabil: Quicksort, Heapsort, Selectionsort

”Ein Algorithmus arbeitet **in-place bzw. in situ**, wenn er außer dem für die Speicherung der zu bearbeitenden Daten benötigten Speicher nur eine konstante, also von der zu bearbeitenden Datenmenge unabhängige Menge von Speicher benötigt. Der Algorithmus überschreibt die Eingabedaten mit den Ausgabedaten.”

Zum Beispiel:

- Bubblesort, Heapsort, Quicksort
- nicht in-situ: Bucketsort, Mergesort

### Quicksort

Beim Quicksort wird in einer rekursiven Subroutine zuerst ein mittleres (*Pivot*-)Element festgelegt. Dann wird in einer Schleife bis  $i > j$  für aufsteigende  $i$  das erste Element  $\geq Pivot$  und für absteigende  $j$  das erste Element  $\leq Pivot$  gesucht. Falls  $i \leq j$  sollen sich das  $i$ -te und  $j$ -te Element tauschen und  $i = i + 1, j = j - 1$  sein. Wenn dann  $i$  oder  $j$  noch nicht die Feldgrenzen überschritten haben soll jeweils für das Teilfeld  $1 : j$  bzw.  $i : n$  der rekursive Aufruf erfolgen.

### Sortieren einer Liste

Eine Firma speichert die Daten ihrer Kunden in einer Text-Datei *leute.dat*.

Diese Liste wird meistens nach Namen sortiert benötigt, jedoch möchte die Firma jedem ihrer Kunden zum Geburtstag gratulieren und benötigt dafür die Liste ab und zu auch nach Geburtstagen sortiert.

Um dies umzusetzen benötigen wir die folgenden zwei Module:

Zuerst brauchen wir das Modul *contmod* mit

- dem abstrakten Datentypen *name*, der zwei Komponenten vom Typ CHARACTER mit maximaler Länge von 15 Zeichen für den Nach- und Vornamen des Kunden enthält.
- dem abstrakten Datentypen *datum*, der drei ganzzahlige Komponenten für Tag, Monat und Geburtsjahr enthält.
- dem abstrakten Datentypen *content*, der eine Komponente vom Type *name* und eine Komponente vom Typ *datum* enthält.

- einem Interface zur Überladung von  $<$  und den zwei implementierenden logischen Funktionen *klnam* und *kldat*, welche zwei Objekte vom Typ *name* und *datum* miteinander vergleichen.

Im zweiten Modul *listmod* benötigen wir außerdem:

- einen abstrakten Datentypen *elem*, welcher eine Komponente vom Typ *content* und vier *elem*-POINTER enthält, welche auf das vorherige bzw. auf das nächste Datum und auf den vorherigen bzw. nächsten Namen zeigen.
- einen ADT *liste*, welcher zwei POINTER, die auf den Kopf der nach Namen bzw. nach Geburtsdatum sortierten Liste zeigen, enthält.
- eine Subroutine *insert*, die ein neues Element in eine sortierte Liste einfügt (nach Namen und Datum)
- die Subroutinen *makeList*, die eine Datei öffnet und mittels *insert* die Liste sortiert aufbaut
- zum Ausgeben der Liste brauchen wir eine Subroutine *writeList*, welche je nach Eingabe *listeByDat* oder *listeByNam* aufruft und nach Datum oder Namen sortiert schreibt

Hinweis: Alle Pointer sollen direkt in der Typdefinition initialisiert werden.

Letztendlich soll noch ein Hauptprogramm zum Testen der Module geschrieben werden. Sind die hier implementierten Sortiervverfahren in-situ und stabil?

### **Zusatz-Aufgabe**

- Schreibe das Unterprogramm für Quicksort. Als Parameter wird ein Feld übergeben (ähnlich zu Übung 7), welches nach Namen oder Datum sortiert werden soll. Füge wieder eine Zählvariable hinzu, welche die Anzahl der zum Sortieren benötigten Schritte mitzählt.