Universität Ulm - Abteilung Angewandte Informationsverarbeitung

1.Übungsblatt (26.04.01 bis 10.05.01)
zur Vorlesung Allgemeine Informatik II für Wirtschaftswissenschaftler

SS 2001

Aufgabe 1 (10 Punkte)

In der Vorlesung wurde der *Quicksort-Algorithmus* vorgestellt. Dieser ist zwar sehr schnell, aber leider nicht ganz so leicht zu verstehen. Eine andere Möglichkeit zum Sortieren von Daten ist der *BubbleSort-Algorithmus*.
Das Prinzip dieses Algorithmus ist relativ einfach. Man vergleicht immer zwei benachbarte Elemente und tauscht ihre Position, wenn sie nicht in der "korrekten" Reihenfolge stehen. Auf diesem Weg erreicht man, daß in jedem "Durchlauf" das größte Element des jeweiligen Teil-Arrays nach vorne geschoben wird.

Schematisch funktioniert der Algorithmus wie folgt:

1. Durchlauf (bis zum vorletzten Element) :	Vergleiche ein Objekt mit dem nächsten und tausche, wenn das zweite kleiner ist. Auf diese Weise läuft die zugehörige Schleife von 0...ArraySize-2.
2. Durchlauf (bis zum drittletzten Element) :	Gleiche Idee wie beim 1. Durchlauf, jedoch läuft die Schleife hier von 0...ArraySize-3.
3. Durchlauf (bis zum viertletzten Element) :	Same Procedure ... hier von 0...ArraySize-4.

... und so weiter ...

Mit diesem Ansatz kann ein Array **aufsteigend** sortiert werden (vgl. Beispiel).

Ihre Aufgabe ist nun, ein Oberon-Programm zu schreiben, das den *BubbleSort-Algorithmus* verwendet. Um es Ihnen ein bißchen leichter zu machen, habe ich das DEFINITION File hier abgelegt.

Nützliche Hinweise:

- Bei der Prozedur **ReadData()** sollten Sie sich an dem Beispiel im Skript (*Quicksort*) orientieren. Achten Sie jedoch darauf, daß als zusätzliche Einschränkung **genau** 10 Zahlen in jedem Array stehen müssen. Ansonsten soll das Programm mit einer Fehlermeldung abbrechen (siehe Beispiele).
  Achten Sie auch darauf, daß Sie bei dieser Prozedur immer einen BOOLEAN-Wert zurückgeben.

Aufgabe 2 (10 Punkte)

Nachdem Sie (hoffentlich!) den *BubbleSort-Algorithmus* problemlos verstanden und programmiert haben, geht es in dieser Aufgabe nun darum, das Programm aus Aufgabe 1 noch um ein nützliches Feature zu erweitern. Der Benutzer soll entscheiden können, ob er seine Daten lieber auf- oder absteigend sortiert haben möchte.

Das Programm soll diese Information (genau wie die Daten) über die Standard-Eingabe erhalten. Als erstes soll die Sortier-Richtung eingelesen werden, danach dann die Daten. Zulässige Eingaben sollen nur **ascending** (aufsteigend) und **descending** (absteigend) sein. In allen anderen Fällen soll das Programm mit einer Fehlermeldung abbrechen.
Außerdem ist es sinnvoll, für dieses Programm die Übergabe von Prozeduren als Parameter (vgl. Skript) zu wiederholen.

Hier finden Sie das DEFINITION File mit dem "öffentlichen" Teil des Programms. Vergessen Sie nicht, daß Sie noch die eine oder andere Prozedur hinzufügen müssen, damit das Programm funktionieren kann.

Auch hier gibt es wieder ein Beispiel, wie das Programm am Ende funktionieren soll.

Nützliche Hinweise:

- Die Abfrage der Sortier-Richtung sollte über eine separate Prozedur gesteuert werden. Ich empfehle den folgenden Prozedur-Kopf:
  ```
  PROCEDURE GetOption(VAR flag : CHAR) : BOOLEAN;
  (* Checks if either "ascending" (-> 'a') or "descending" (->'d') have been
  submitted, otherwise the return value is FALSE *)
  ```
- Zur Variation der Sortierreihenfolge sollten Sie sich überlegen, welcher Teil in der Prozedur **BubbleSort()** aus Aufgabe 1 verallgemeinert werden muß. Genau diesen Teil sollten Sie dann durch die Übergabe von entsprechenden Prozeduren steuern.
- Die Prozeduren **ReadData()** und **WriteData()** können bei Verwendung der empfohlenen Struktur 1:1 aus Aufgabe 1 übernommen werden.

Viel Glück!!!