OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

4.2 Übung Informatik PG 1

Diese Übung kombiniert Arrays, Sortieralgorithmen und Rekursion.

In der Vorlesung haben wir uns Bubblesort angeschaut, einen recht einfachen aber auch langsamen Sortieralgorithmus. Neben Bubblesort gibt es eine Vielzahl weiterer Sortieralgorithmen, zum Beispiel *Quicksort*.

Der Quicksortalgorithmus setzt das Prinzip "Teile und Herrsche" ein, das heißt, dass eine schwierige Aufgabe so lange in Teilschritte zerlegt wird, bis ein einfach auszuführender Teilschritt erreicht wird. Konkret bedeutet das, dass die Arrayelemente immer wieder in zwei Teile zerlegt und diese dann separat sortiert werden. Implementiert werden kann dieses Verhalten am einfachsten per Rekursion.

Eine Erklärung zur Funktionsweise von Quicksort finden Sie in den Folien.

Aufgabe: Implementieren Sie den Quicksort-Algorithmus für ein Array mit 10 Werten

- Schreiben Sie eine Funktion namens *Partition*, die das übergebene Array wie in den Folien beschrieben durchsucht. Partition übernimmt das Array sowie den Index des ersten und letzten Arrayelements und gibt den Index des neuen Pivotelements zurück. Inhalt der Funktion:
    - Der Arraysuchlauf wird ausgeführt, bis der rechte Index **kleiner oder gleich** dem linken Index ist
    - Das letzte Element im Array ist das **Pivotelement**
    - Der **linke Index** läuft aufsteigend vom ersten Element durch das Array, bis ein Wert gefunden wird, der **größer** als das Pivotelement ist
    - Der **rechte Index** läuft absteigend vom vorletzten Arrayelement durch das Array, bis ein Wert gefunden wird, der **kleiner** als das Pivotelement ist
    - Überlegen Sie sich was beim Erhöhen bzw. Verringern der Indizes passiert, wenn linker und rechter Index auf einem Wert **gleich** dem Pivotelement landen und wie Sie dieses Verhalten umgehen können (Stichwort do-while)
    - Wenn rechter und linker Index stehen geblieben sind, werden die Werte mittels Ringtausch vertauscht
    - Ist der Durchlauf durch das Array beendet, wird das Pivotelement mit dem äußersten linken Element der rechten Seite getauscht (bedenken Sie, dass der Durchlauf dann beendet wird, wenn der rechte Index kleiner oder gleich dem linken Index ist - welcher Index steht dann auf der äußersten linken Stelle der rechten Seite?)
    - Abschließend wird der neue Index des Pivotelements zurückgegeben
    - Die Partition-Funktion wird nicht in der main-Funktion aufgerufen, sondern nur in der Quicksort-Funktion
- Schreiben Sie eine weitere Funktion namens *Quicksort*, die die Partition-Funktion aufruft und anschließend sich selbst zweimal, einmal für das linke Subarray und einmal für das rechte. Wird die

Quicksort-Funktion in main aufgerufen, so übernimmt sie das Array sowie den Index des ersten und des letzten Arrayelements.

  - Überlegen Sie sich, in welchen Fällen die Quicksort-Funktion sich selbst aufrufen soll (Hinweis: Indizes)
  - Bei den zwei aufeinanderfolgenden, rekursiven Aufrufen innerhalb von Quicksort übernimmt der erste rekursive Aufruf das untere Subarray (also vom ersten Arrayelement bis unter den Index des Pivotelements (also ohne das Pivotelement)) und der zweite rekursive Aufruf das obere Subarray:

```
pivotIndex = partition(array, low, high);
quicksort(array, low, pivotIndex - 1);
quicksort(array, pivotIndex + 1, high);
```

- Drucken Sie das Array unsortiert, die Subarrays sowie sortiert.
- Testen Sie auch für Grenzfälle (alle Werte kleiner Pivot, alle Werte größer Pivot, alle Werte 0, ...) mittels der assert-Funktion