Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Andreas Hildebrandt

3. Übung zur Vorlesung Programmierung II, SS 2003

Aufgabe 1: QuickSort
Ein häufig verwendetes Sortierverfahren ist der sogenannte *Quicksort-Algorithmus*. Dieser besteht aus zwei wesentlichen Schritten:

- Umsortieren des Arrays $V = (v_p, \ldots, v_r)$, so dass zwei nichtleere Teilarrays $V_1 = (v_p, \ldots, v_q)$ und $V_2 = (v_{q+1}, \ldots, v_r)$ entstehen mit der Eigenschaft: $\forall v_i \in V_1, \forall v_j \in V_2 : v_i \leq v_j$ Dazu wird v_p als sogenanntes *Pivotelement* gewählt, d.h. das Array V wird so in die Teilarrays V_1 und V_2 umsortiert, dass gilt: $\forall v_i \in V_1 : v_i \leq v_p, \forall v_j \in V_2 : v_j > v_p$.
- Rekursives Sortieren der Arrays V_1 und V_2.

Implementieren Sie Quicksort für den Datentyp `vector<int>`.

Aufgabe 2: Klassen, Teil 1
Implementieren Sie eine Klasse `Vorlesungsteilnehmer`, welche die folgende Funktionalität enthält:

- In der Klasse können der Name, die Matrikelnummer und das Studienfach des Teilnehmers gespeichert werden
- Alle diese Eigenschaften können in einem Konstruktor übergeben werden. Außerdem gibt es einen Defaultkonstruktor, der jede dieser Eigenschaften mit einem sinnvollen Standardwert belegt.
- Jede der oben genannten Eigenschaften kann mittels jeweils einer Methode der Klasse ausgelesen, mittels einer anderen gesetzt werden.
- Die Klasse enthält einen `vector<int>`, in welchem die erreichte Punktzahl für jedes Übungsblatt gespeichert werden kann. Auf diese Daten kann wiederum mittels zweier Funktionen zugegriffen werden: eine gibt die erreichte Punktzahl für ein bestimmtes Übungsblatt zurück, die zweite setzt sie.
- Die Klasse enthält eine Funktion `bool bestanden()`, die überprüft, ob eine gewisse Punktzahl – welche über die Funktion `void bestehensgrenze(int punkte)` gesetzt werden kann – schon überschritten wurde.

Aufgabe 3: Klassen, Teil 2
Implementieren Sie eine Klasse `myIntegerVector`, welche die folgende Funktionalität enthält:

- Die Klasse enthält einen `vector<int>`. Dieser kann in einem Konstruktor übergeben werden, im Defaultkonstruktor wird ein leerer `vector<int>` erzeugt.
- Die Klasse enthält eine Methode `void zufallswerte(int n)`, die die Länge des `vector` auf n setzt und ihn anschließend mit Zufallszahlen füllt.
- Die Klasse enthält eine Methode `elementAnfuegen(int e)`, die hinten an den `vector` das Element e anfügt.
- Die Klasse enthält eine Methode `int& element(int n)`, die eine Referenz auf das n-te Element zurückliefert, sowie eine Mehode `const int& element(int n) const` für konstante Referenzen.
- Die Klasse enthält eine Methode, welche den `vector` mittels InsertionSort, MergeSort oder QuickSort sortiert. Die Wahl des Sortierverfahrens kann der Benutzer wiederum über eine weitere Methode treffen.
- Die Klasse enthält eine Methode, die den `vector` nach einem gegebenen Element e binär durchsucht. Dabei soll der `vector` vorher – nur falls dies notwendig sein sollte – sortiert werden.
- Als Zusatzaufgabe können Sie – genügend Motivation vorausgesetzt – schließlich die Klasse `myIntegerVector` in eine Template-Klasse mit Namen `myVector` umwandeln.
