

Programmieren 1 Übung: Einfache C++-Konstrukte

Klaus Kusche

1.) Rangliste sortieren und ausgeben

Schreib ein Programm,

- das mit ein oder mehreren double-Werten auf der Befehlszeile aufgerufen wird (die Werte sollen Ergebnis-Zeiten eines Bewerbes darstellen und sind in Startnummern-Reihenfolge angegeben),
- diese Werte zuerst einmal in ein Array einliest und sich dabei zu jedem Wert die Position in der Eingabe (= Startnummer) merkt,
- dieses Array dann aufsteigend nach Zeiten sortiert,
- und schließlich wahlweise die Ausgabe der gesamten Ergebnis-Liste oder die Abfrage einzelner Startnummern erlaubt:
 - Dazu soll das Programm den Benutzer zuerst einmal fragen, ob die Gesamtliste ('g') oder der Platz einer einzelnen Startnummer ('s') ausgegeben werden soll,
 - im Falle einer Startnummer den Benutzer auch nach der auszugebenden Startnummer (nicht Rangnummer in der sortierten Ausgabe!) fragen,
 - und dann die gewünschte Ausgabe liefern.

Das wird wiederholt, bis der Benutzer bei der Auswahl nicht 'g' / 's' eingibt (groß- oder kleingeschieben, verwende **tolower**); dann endet das Programm.

Wir wollen mit diesem Beispiel C++-Konstrukte üben. Daher:

- Verwende zur Ein- und Ausgabe die Funktionalität von **iostream**, nicht stdio.h. Wohin gehören Fehlermeldungen?
- Verwende auch sonst **nur C++-Header** (bzw. die C++-Variante von C-Headern).
- Verwende für das Speichern der Startnummer und der Zeit eines Teilnehmers eine **Struktur**. Verzichte dabei überall, wo es möglich ist, auf das "**struct**".
- Die Nummern und Zeiten aller Teilnehmer sind in einem Array solcher Strukturen zu speichern.
Dieses Array soll dynamisch genau in der benötigten Größe angelegt werden.

Verwende zum Anlegen und Freigeben des Arrays der Startnummern und Zeiten **new** und **delete**.

Details zu **new** und **delete** lernen wir erst später, deshalb hier kurz vorab:

- "**new xxx[anz]**" legt ein neues Array mit **anz** Elementen vom Typ **xxx** an und liefert einen Pointer (vom Typ **xxx ***) auf dessen Anfang als Ergebnis zurück.
- Wenn **p** ein Pointer auf ein mit **new** angelegtes Array ist, gibt "**delete [] p;**" das Array wieder frei.

- Das Sortieren implementieren wir diesmal selbst, ganz primitiv mit Bubblesort. Verwende beim Sortieren eine **selbstgeschriebene Funktion tausche**, die zwei Teilnehmer-Strukturen als Referenz (nicht Pointer!) übergeben bekommt und die Inhalte der beiden Strukturen vertauscht.
- Auch das Ausgeben soll über eine selbstgeschriebene Funktion erfolgen, und zwar in beiden Fällen mit derselben Funktion, die je nach Aufruf Verschiedenes leistet:
 - Wird sie mit 2 Parametern (Array der Ergebnisse und Anzahl der Einträge) aufgerufen, gibt sie alle Einträge des Arrays in sortierter Reihenfolge zeilenweise aus.
 - Wird sie zusätzlich mit einem 3. Parameter (Startnummer) aufgerufen, geht sie auch das gesamte Array durch, gibt aber nur den einen Eintrag mit dieser Startnummer aus (oder gar keinen, falls die Startnummer nicht vorkommt).

Tipp: Sorge dafür, dass der dritte Parameter der Funktion automatisch den Wert 0 bekommt, wenn die Funktion mit nur 2 Parametern aufgerufen wird, und reagiere im Code der Funktion entsprechend auf diesen Wert:
Bei 0 wird jeder Eintrag ausgegeben, bei ungleich 0 nur der Eintrag mit genau dieser Startnummer.