FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES

PRG1 - Hausaufgabe 05

Thema: Arrays, Sortieralgorithmen, Zeitkomplexität

Es sollen verschiedene Sortieralgorithmen für Arrays als Funktionen realisiert und in ihrem Zeitverhalten gegenübergestellt werden..

Ziel ist die Übung der Umsetzung theoretisch fundierter Algorithmen eine anschauliche Illustration von komplexitätstheoretischen Zusammenhängen.

Vorbereitungen

- Anlegen eines neuen, leeren C++-Projekts.
- Legen Sie in dem Projekt eine Datei ***main.cpp*** an und erstellen Sie darin das Grundgerüst einer *main*-Routine.
- Erzeugen Sie für die in den Teilaufgaben geforderten Sortierfunktionen ein Dateipaar (jeweils Header- und Code-Datei) mit dem Namen „sorting“ (***sorting.h*** + ***sorting.cpp***).
- Erstellen Sie bei Bedarf für den zu implementierenden Rahmenalgorithmus weitere Dateipaasre für die Hilfsfunktionen.

Aufgaben

Nr.	Beschreibung	Kat. / Punkte
1.	Schreiben Sie eine Funktion `isSorted` in C, die einen vorgegebenen Teilbereich eines Arrays über int-Werten auf Sortierheit bezüglich der Relation ≤ prüft. Parameter sollen das Array, der Anfangsindex des Bereichs und der auf das Ende des Bereichs folgende Index (kann unmittelbar außerhalb des Arrays liegen) sein. Der Rückgabetyp soll einen Wahrheitswert beinhalten.	★ 2
2.	Schreiben Sie eine Funktion `bubbleSort`, die einen vorgegeben Teilbereich eines int-Arrays nach dem Algorithmus der Blasensortierung bezüglich der Relation ≤ ordnet. Die Parameterstruktur sei die gleiche wie in Teilaufgabe 1.	★★ 4
3.	Schreiben Sie eine Funktion `selectionSort`, die einen vorgegeben Teilbereich eines int-Arrays nach dem Algorithmus der Sortierung durch Auswahl bezüglich der Relation ≤ ordnet. Die Parameterstruktur sei die gleiche wie in Teilaufgabe 1.	★★ 4
4.	Schreiben Sie eine Funktion `insertionSort`, die einen vorgegeben Teilbereich eines int-Arrays nach dem Algorithmus der Sortierung durch Einfügen bezüglich der Relation ≤ ordnet. Die Parameterstruktur sei die gleiche wie in Teilaufgabe 1.	★★ 5
5.	Schreiben Sie eine Funktion `quickSort`, die einen vorgegeben Teilbereich eines int-Arrays nach einer möglichst guten Variante des QuickSort-Algorithmus bezüglich der Relation ≤ ordnet. Die Parameterstruktur sei die gleiche wie in Teilaufgabe 1.	★★★ 10

Nr.	Beschreibung	Kat. / Punkte
6.	Schreiben Sie einen Testrahmen, der ein int-Array zunächst festgehaltener Größe (zum Beispiel 20) jeweils zunächst monoton wachsenden, dann monoton fallenden und schließlich mehrmals (z. B. 10mal) mit zufällig erzeugten Werten füllt und jedesmal unter automatischer (d. h. programmrealisierter) Zeitnahme durch alle verfügbaren Sortieralgorithmen sortieren lässt. Je Sortieralgorithmus (und Array-Größe) sind von den erhaltenen Zeiten der zufälligen Füllungen der Mittelwert, der Minimal- und der Maximalwert zu bestimmen und neben den Zeiten für vorsortiertes und umgekehrt vorsortiertes Array zu einem Verbund zusammengestellt aufzuheben (in einem entsprechenden Array, indiziert nach Sortieralgorithmus) und tabellarisch auszugeben. Dieser Vorgang ist nun mit in bestimmten Schritten (Vorschlag: Verdoppelung) wachsender Arraygröße zu wiederholen. Die Einzelergebnisse je Größe des zu sortierenden Arrays sind wieder zu einem Array (Indexbereich Arraygrößenschritte) zusammengefasst zu speichern. Hinweis: Sie müssen nicht unbedingt verschieden große int-Arrays erzeugen, sondern können ein int-Array mit Maximalgröße (letzte und höchste Testgröße – diese sollte mindestens in der Größenordnung 10^5 liegen) reservieren und dann nur mit Teilfüllungen wachsender Größe arbeiten.	★★★ 10
7.	Dokumentieren Sie in geeigneter Form die bei Ihren Tests erhaltenen Zeiten und begründen Sie, z. B. mittels verschiedener Quotientenbildungen zu berechneten Kandidatenfunktionen (linear, quadratisch, $n \cdot \log n$), welche mathematischen Abhängigkeiten von der Arraygröße sich daraus plausibel ableiten lassen.	★★★ 4
8. *	(Freiwillige Zusatzaufgabe) Binden Sie die Turtleizer-Bibliothek (siehe E-Learning-Portal) in Ihre Projektmappe ein, zeichnen Sie auf der Turtle-„Spielwiese“ ein Koordinatensystem geeigneter Größe und Achsenteilung (Abszisse: Array-Größe, Ordinate: Abarbeitungszeit) und tragen Sie darin zunächst die Verläufe der Funktionen n, n^2 und $n \cdot \mathrm{ld}\, n$ farbig ein. Tragen Sie dann mit anderen Farben die erhaltenen Ergebnisse für die einzelnen Algorithmen als geradlinig interpolierte Funktionsverläufe dazu.	★★★ 10