LU01.A03 - myBubbleSort

Rahmenbedingungen

- Sozialform: Einzelarbeit
- Hilfsmittel: Openbooks
- Zeit: 70 Minuten
- Erwartetes Resultat: JavaScript-File, das eine Reihe von Zahlen und Buchstaben der Reihen nach, aufsteigend oder absteigend, sortieren kann.

Hinweise

Wichtig bei der Umsetzung Ihrer Lösung ist, dass sie nach best-practise programmieren. Konkret heisst das:

- Ihr Script, sowie die verwendeten Subroutinen (Funktionen, Methoden) müssen ei-nen Header haben: Autor, Datum, Angaben zur Input-, und Output-Parameter, eine kurze Beschreibung zum Verhalten der Funktion.
- Bei fehlenden Parametern wird die eingebende Person entsprechend über das HTML-Formular oder der Kommandozeile informiert.
- Verwenden Sie die unbedingt die passenden Datentypen (primitive oder komplexe).
- Verwenden Sie keine JS-Bibliotheken, sondern programmieren Sie alle relevanten Funktionen selbst. Letztendlich ist das das eigentliche Ziel dieser Übung.
- Aus Gründen der Datenkapselung und der umsichtigen Programmierung realisieren sie Ihre Funktionen ausschliesslich mit Inputparameter und Return-Values. Es wird also möglichst nicht auf globale Werte innerhalb der Methoden zugegriffen.
- Die für die Berechnung benötigten Werte können über Commandline, als Variable oder über ein simples HTML-Formular eingegeben werden.
- Verschwenden Sie keine Zeit in eine *schöne“ Oberflächengestaltung*, weil es aktuell um Programmier-Praxis, und nicht um Gestaltung geht.

Auftrag

Sortierungen finden wir überall in unserem Alltag. Beispielsweise sind unsere Kontakte alphabetisch sortiert, in der Regel aufsteigend. Der **BubbleSort-Algorithmus** ist ein solcher Sortieralgorithmus, der sehr gut zu Schulungszwecken verwendet werden kann.

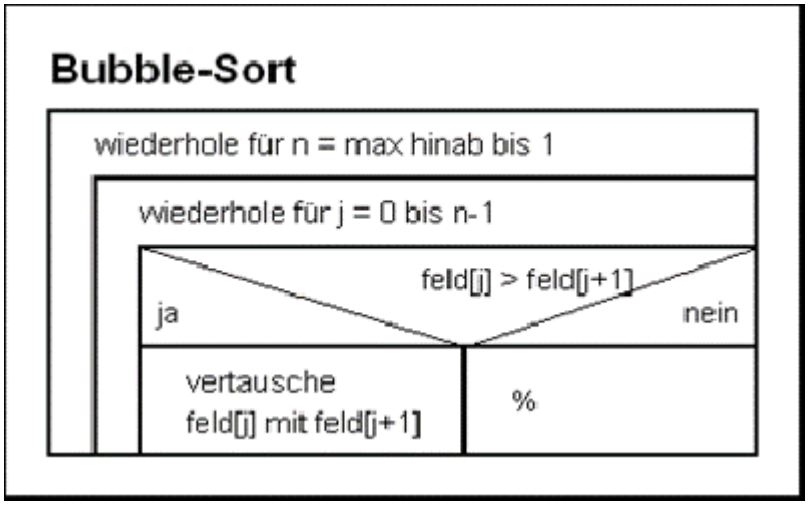
Studyflix: Bubblesort

Bubblesort-Aufgabe

Nachfolgend finden Sie die Struktogramme von zwei Varianten des Sortier-Algorithmus

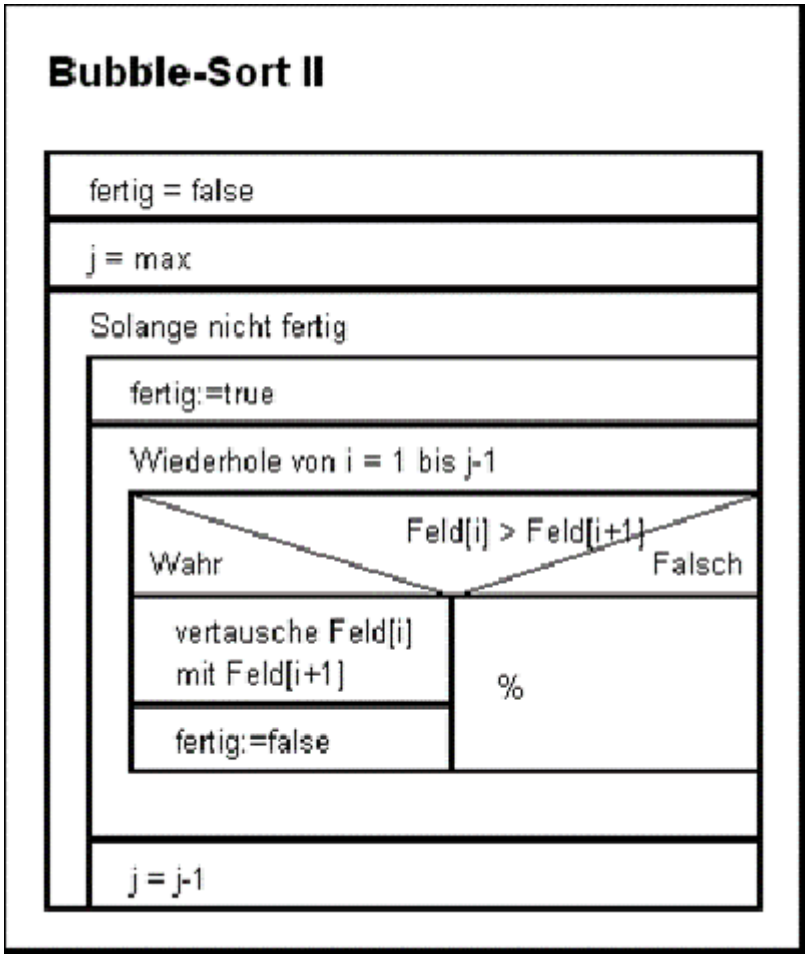
Teilauftrag 1: Statisch

In zwei FOR-Schleifen wird ein unsortiertes Array eine feste Anzahl mal durchlaufen. Die Anzahl der Schleifendurchlaeufe ist abhaengig von der Anzahl unsortierter Array-Elemente.



Teilauftrag 2: Flag

Variante 2 arbeitet intelligenter als die Variante 1. Mittels eines *Flag* (Ampel) wird bei jedem Durchlauf des Arrays ermittelt, ob die korrekte Sortierung vorliegt. Falls nicht wird ein weiterer Durchlauf gestartet.



Teilauftrag 3: Benchmarking

Um die Effizienz der Vertauschungen beider Varianten miteinander vergleichen zu können, brauchen wir eine Art *Benchmarking*, also einen Tausch- und einen Schleifen-Zähler, welche an der passenden Stelle im Code eingefügt werden müssen.

1. Der Tausch-Zähler wird inkrementiert (hochgezählt), sobald zwei Zahlen vertauscht werden mussten, weil Sie in der falschen Reihenfolge vorlagen.
2. Der Schleifenzähler wird in der inneren Schleife gesetzt, sodass der die Anzahl Schleifen-Durchläufe mitzählt.

Teilauftrag 4: Wahlweise Auf- oder absteigend sortiert

Passen Sie Ihre Algorithmen so an, sodass die beim Funktionsaufruf festelegen können, ob auf oder absteigend sortiert werden muss. Dazu müssen Sie einen zusätzlichen Parameter mitgegen.

Lösungen

LU01.L03

Volkan Demir

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m307/learningunits/lu01/aufgaben/03?rev=1772099208

Last update: **2026/02/26 10:46**