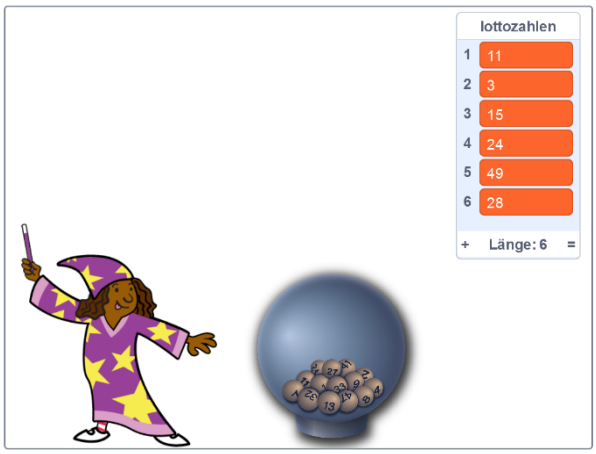
SORTIEREN: BUBBLE SORT



Auf digitalen Geräten (Smartphone, Computer, etc.) sucht man ständig etwas: Eine App, einen Song, ein Video. Auch auf Webseiten und innerhalb von Apps ist man ständig auf der Suche. Denken Sie nur mal daran, was Sie als erstes tun, nachdem Sie die Amazon-Webseite geöffnet haben.

So eine Suche kann aber nur dann schnell gehen, wenn die Daten, die man durchsucht, *sortiert* sind! Nicht nur Menschen, sondern auch Computer finden das Gesuchte in sortierten Daten wesentlich schneller. Wir werden uns deshalb anschauen, wie man man das hinkriegt: Wir sortieren Daten!

Bubble Sort: Leicht zu verstehen!

In der Praxis verbringen viele Computer einen Großteil ihrer Zeit damit, Daten zu sortieren. Weil das Sortieren so wichtig ist, hat man viele verschieden *Algorithmen* (also klar definierte Vorgehensweisen) erfunden, um dieses Ziel zu erreichen. Wir schauen uns jetzt einen Sortier-Algorithmus an, der ziemlich leicht zu verstehen ist: *Bubble Sort*.

Der Größte-Zahl-ans-Ende-Algorithmus

- Wir fangen links an und vergleichen zwei benachbarte Elemente des Arrays.
- Wenn das linke Element größer ist als das rechte, dann „wandert es nach rechts“, es wird also getauscht.
- Nachdem wir so alle benachbarten Elemente des Arrays verglichen haben, steht das größte Element am Ende des Arrays!

Beispiel: Wir haben ein Array mit 4 Zahlen

Durchgang 1

array[1]	array [2]	array [3]	array [4]
6	4	9	2
6 > 4 ? Ja → Tausch			

array [1]	array [2]	array [3]	array [4]
4	6	9	2
	6 > 9 ? Nein → KeinTausch		

array [1]	array [2]	array [3]	array [4]
4	6	9	2
		9 > 2 ? Ja → Tausch	

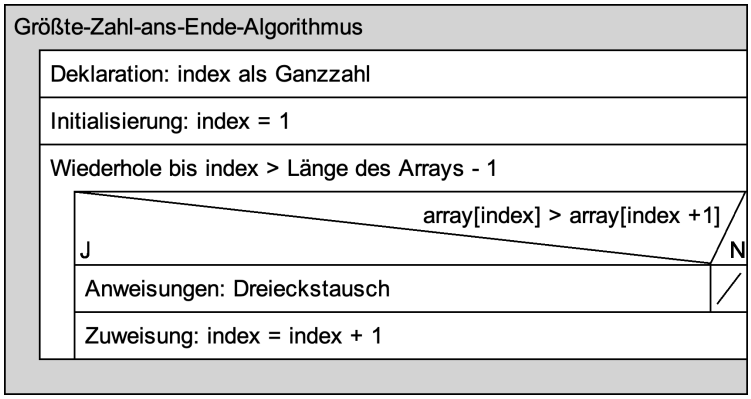
array [1]	array [2]	array [3]	array [4]
4	6	2	9

Das größte Element steht jetzt am Ende!

Wie oft muss verglichen werden?
Wir sehen, dass wir bei 4 Zahlen genau 3 Vergleiche benötigen. Daraus können wir diese Erkenntnis gewinnen:

Anzahl Vergleiche = Länge des Arrays - 1

Struktogramm: Größte-Zahl-ans-Ende



Aufgabe:

1. Öffnen Sie das Projekt *7200-GroesstesElementAnsEnde-AUFGABE.* Entwerfen Sie in Scratch den Programmcode, mit dem Sie das größte Element ans Ende des Arrays verschieben!

Bubble Sort: Und wie geht es jetzt weiter?

Unser Ziel ist es, das komplette Array zu sortieren. Genau das kriegen wir hin, wenn wir den *Größte-Zahl-ans-Ende-Algorithmus* in mehreren Durchgängen auf das restliche Array anwenden. Schauen wir uns das im Beispiel an:

array [1]	array [2]	array [3]	array [4]
4	6	2	9
4 > 6 ? Nein → Kein Tausch			

Rest-Array: Dieser Teil ist noch nicht sortiert!

array [1]	array [2]	array [3]	array [4]
4	6	2	9
	6 > 2 ? Ja → Tausch		

array [1]	array [2]	array [3]	array [4]
4	2	6	9

Durchgang 2

Das größte Element im restlichen Array wurde ans Ende verschoben

Damit haben wir es fast geschafft: Noch ein weiterer Durchgang mit dem *Größte-Zahl-ans-Ende-Algorithmus*:

Durchgang 3

array [1]	**array [2]**	array [3]	array [4]
4	2	6	9
4 > 2 ? Ja → Tausch			

array [1]	array [2]	array [3]	array [4]
2	4	6	9

Bubble Sort: Wie viele Durchgänge brauchen wir?

Sie sehen: Wenn wir unseren *Größte-Zahl-ans-Ende-Algorithmus* mehrmals anwenden, haben wir am Schluss ein komplett sortiertes Array!

Die Frage ist jetzt: **Wie viele Durchgänge brauchen wir, bis das Array sortiert ist?**

Fassen wir unsere Beobachtungen zusammen:

- Unser Array hatte **4** Elemente!
- Wir haben **3** Durchgänge gebraucht!

➔ Vermutung: Die Anzahl der Durchgänge entspricht ***Länge des Arrays – 1***

Kann das sein? In jedem Durchgang verschieben wir die jeweils größte Zahl ans Ende des übrig gebliebenen Arrays. Dann müssten es doch genauso viele Durchgänge sein, wie Elemente im Array vorhanden sind!

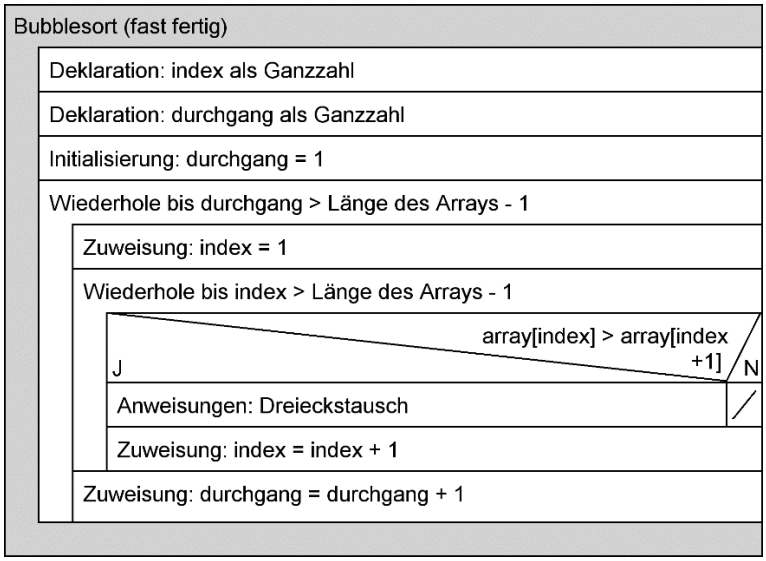
ABER:
Im letzten Durchgang (im Beispiel: Durchgang 3) werden gleich **2** Elemente richtig einsortiert!

⇨ Unsere Vermutung stimmt!

Wir müssen also das Folgende tun, um ein Array mit *Bubble Sort* zu sortieren:

```
Bubblesort (fast fertig)
    Deklaration: durchgang als Ganzzahl
    Initialisierung: durchgang = 1
    Wiederhole bis durchgang > Länge des Arrays - 1
        Größte-Zahl-ans-Ende-Algorithmus
        Zuweisung: durchgang = durchgang + 1
```

Der Teil, der wiederholt werden soll, muss also ebenfalls in eine Schleife verschoben werden:

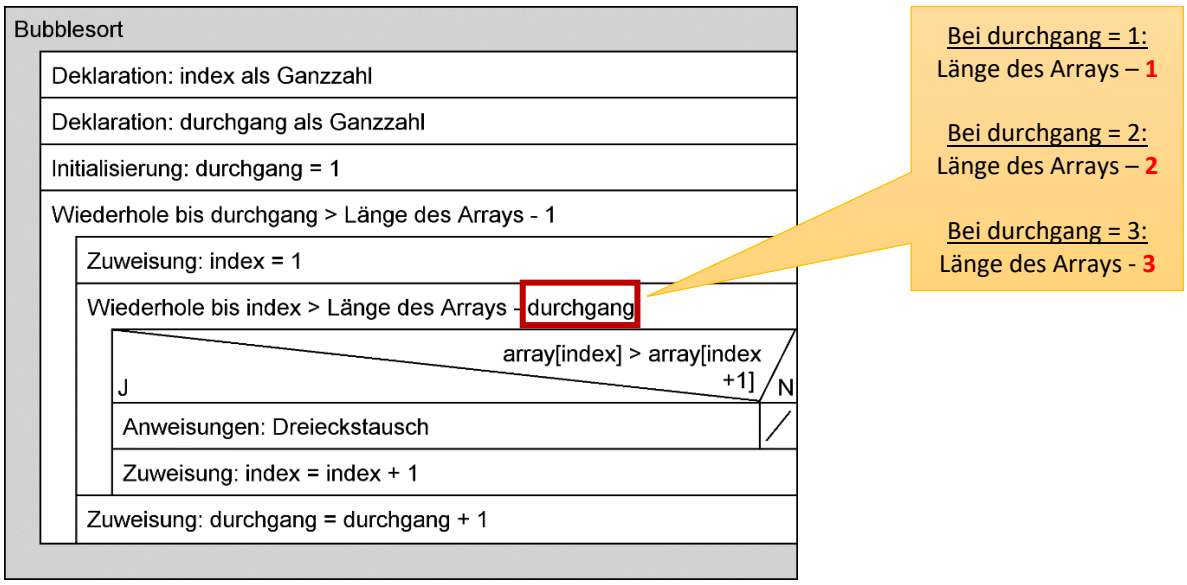


Eine Schleife in einer Schleife! Das sieht ja jetzt echt kompliziert aus. Bevor wir uns genauer anschauen, wie das funktioniert, müssen wir erst noch einen kleinen „Schönheitsfehler“ beseitigen, der uns bisher noch nicht aufgefallen ist!

Betrachten wir nochmal, welche Elemente wir in den einzelnen Durchgängen miteinander vergleichen:

Durchgang	**Elemente, die verglichen werden**	**<u>Größte-Zahl-ans-Ende-Algorithmus</u> array[index] > array[index + 1] ?**
1	1 und 2 2 und 3 3 und 4	index ist 1: array[1] > array[2] ? index ist 2: array[2] > array[3] ? index ist 3: array[3] > array[4] ?
2	1 und 2 2 und 3	index ist 1: array[1] > array[2] ? index ist 2: array[2] > array[3] ?
3	1 und 2	index ist 1: array[1] > array[2] ?

Sie müssten sehen, dass die Schleife zum Vertauschen der Elemente jedes Mal kürzer läuft. Wir verändern unser Struktogramm entsprechend, um das zu berücksichtigen:



Eine Schleife in einer Schleife: Wie geht das?

Äußere Schleife
durchgang = **1**

Innere Schleife
Wird komplett durchlaufen bis index > Länge des Arrays – **1**

Äußere Schleife
durchgang = **2**

Innere Schleife
Wird komplett durchlaufen bis index > Länge des Arrays – **2**

….

⇨ Die innere Schleife läuft schneller als die äußere!

Aufgabe:

2. Sie haben eine Namensliste, die sortiert werden soll:

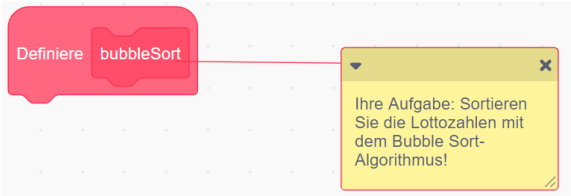
Eva	Anja	Dragan	Jasmin	Alex

Sortieren Sie diese Liste „mit Hand“, indem Sie den Bubble Sort-Algorithmus verwenden. Verwenden Sie dafür diese Vorlage:
7200_Aufgabe2_NamenSortieren.docx

Eva	Anja	Dragan	Jasmin	Alex	
					Durchgang 1
					Durchgang 2
					Durchgang 3
					Durchgang 4

(Hinweis: Sie müssen nicht notieren, ob getauscht wird oder nicht!)

3. Öffnen Sie das Projekt *7210_BubbleSort-AUFGABE.* Ihre Aufgabe ist es, Lottozahlen zu sortieren. Setzen Sie dazu das *Bubble Sort*-Struktogramm um!



4. Öffnen Sie das Projekt *7215_Blumenladen-AUFGABE*. In diesem Projekt wurde bereits ein Array mit dem Namen *verkaeufe* erstellt und gefüllt.

Ihre Aufgabe ist es, das Array *verkaeufe* zu sortieren. Setzen Sie dazu den *Bubble Sort*-Algorithmus ein!