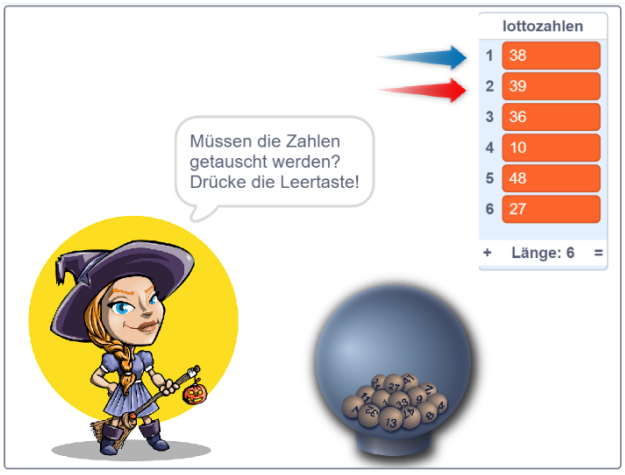
SORTIEREN: SELECTION SORT



Dass das Sortieren auf digitalen Geräten etwas wichtiges ist, wissen wir ja schon. Deshalb suchen Informatiker nach immer besseren Methoden, Werte möglichst schnell zu sortieren und haben sich viele verschiedene Algorithmen ausgedacht, mit denen man das erreichen kann.

Vorher hatten wir den Algorithmus *Bubble Sort* kennen gelernt. Heute schauen wir uns einen Algorithmus an, der ganz ähnlich funktioniert: **Selection Sort**. Hier werden aber (anders als bei Bubble Sort) nicht die benachbarten Elemente miteinander verglichen.

Selection Sort

Der Algorithmus von Selection Sort basiert darauf, dass man sich zuerst das kleinste Element sucht, dann das zweitkleinste und so weiter.

Vorbereitung: Der Kleinste-Zahl-an-den-Anfang-Algorithmus

- Wir vergleichen nacheinander jedes Element mit dem ersten Element des Arrays.
- Wenn wir ein Element finden, das kleiner ist als das erste, vertauschen wir beide.
- Wenn wir das Array bis zum Ende durchlaufen haben, steht das kleinste Element am Anfang!

Beispiel: Wir haben ein Array mit 4 Zahlen

Durchgang 1

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

4 < 6 ? Ja → Tausch

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

9 < 4 ? Nein → KeinTausch

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

2 < 4 ? Ja → Tausch

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

Das kleinste Element steht jetzt am Anfang!

<u>Welche Elemente werden verglichen?</u>

Wir haben folgende Elemente des Arrays miteinander verglichen:

1 und **2**
1 und **3**
1 und **4**

*Um diese Vergleich durchzuführen, brauchen wir also eine Variable, die nacheinander die Werte **2, 3 und 4** annimmt! Wir verwenden hierfür eine Variable index, die wir in jedem Schleifendurchlauf nach oben zählen.*

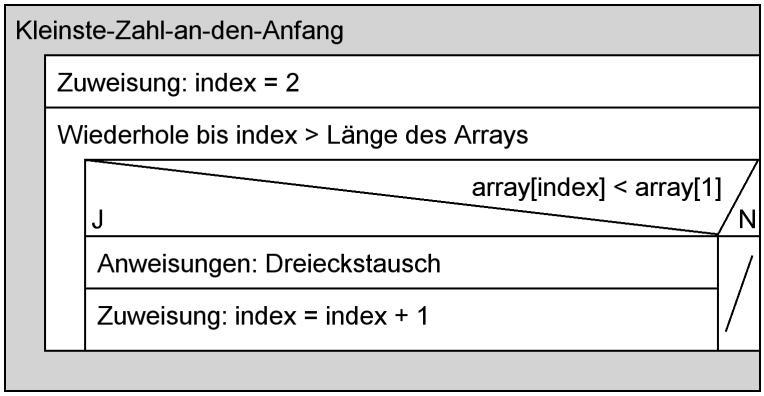
Die Idee ist also:

Setze index auf 2
Wiederhole bis index > 4
….

Weil nicht jedes Array 4 Elemente hat, können wir unsere Idee allgemeiner formulieren:

Setze index auf 2
*Wiederhole bis index > **Länge des Arrays***
….

<u>Struktogramm: Kleinste-Zahl-an-den-Anfang</u>



<u>Aufgabe:</u>

1. Öffnen Sie das Projekt *7300_KleinstesElementAnDenAnfang-AUFGABE.*
Entwerfen Sie in Scratch den Programmcode, mit dem Sie das kleinste Element an den Anfang des Arrays verschieben!

Selection Sort: Und wie geht es jetzt weiter?

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

Durchgang 2

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

Rest-Array: Dieser Teil ist noch nicht sortiert!

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

9 < 6 ? Nein → Kein Tausch

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

4 < 6 ? Ja → Tausch

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

Das kleinste Element im restlichen Array wurde nach vorne verschoben

Damit haben wir es fast geschafft: Noch ein weiterer Durchgang mit dem Kleinste-Zahl-an-den-Anfang-Algorithmus:

Durchgang 3

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

6 < 9 ? Ja → Tausch

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

Selection Sort: Wie viele Durchgänge brauchen wir?

Sie sehen: Wenn wir unseren *Kleinste-Zahl-an-den-Anfang-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 kleinste Zahl an den Anfang 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 *Selection Sort* zu sortieren:

Selection Sort (fast fertig)	
Initialisierung: durchgang = 1	
Wiederhole bis durchgang > Länge des Arrays - 1	
	Kleinste-Zahl-an-den-Anfang-Algorithmus
	Zuweisung: durchgang = durchgang + 1

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

Selection Sort (fast fertig)
- Initialisierung: durchgang = 1
- Wiederhole bis durchgang > Länge des Arrays - 1
  - Zuweisung: index = 2
  - Wiederhole bis index > Länge des Arrays
    - array[index] < array[1] (J / N)
      - J: Anweisungen: Dreieckstausch
      - J: Zuweisung: index = index + 1
      - N: /
  - Zuweisung: durchgang = durchgang + 1

Kleinste-Zahl-an-den-Anfang-Algorithmus

Achtung! Hier stimmt was nicht!

Irgendwie kann das aber so nicht stimmen! Schauen wir uns nochmal an, welche Elemente wir in den einzelnen Durchgängen verglichen haben:

durchgang ist 1	durchgang ist 2	durchgang ist 3
array**[2]** < array**[1]** ? array**[3]** < array**[1]** ? array**[4]** < array**[1]** ?	array**[3]** < array**[2]** ? array**[4]** < array**[2]** ?	array**[4]** < array**[3]** ?

durchgang ist **1**: index beginnt bei **2**
durchgang ist **2**: index beginnt bei **3**
durchgang ist **3**: index beginnt bei **4**
⇨ Index beginnt bei **durchgang + 1**

Der in der Variable *durchgang* gespeicherte Wert gibt an, an welcher Position das kleinste Element gespeichert wird!

Selection Sort
- Zuweisung: durchgang = 1
- Wiederhole bis durchgang > Länge des Arrays - 1
  - Zuweisung: index = durchgang + 1
  - Wiederhole bis index > Länge des Arrays
    - array[index] < array[durchgang] (J / N)
      - J: Anweisungen: Dreieckstausch
      - J: Zuweisung: index = index + 1
      - N: /
  - Zuweisung: durchgang = durchgang + 1

Aufgabe:

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

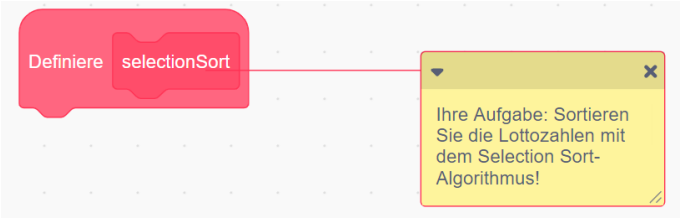
Eva	Anja	Dragan	Jasmin	Alex

Sortieren Sie diese Liste "von Hand", indem Sie den Selection Sort-Algorithmus verwenden. Verwenden Sie dafür diese Vorlage: *7200_Aufgabe2_NamenSortieren.docx*

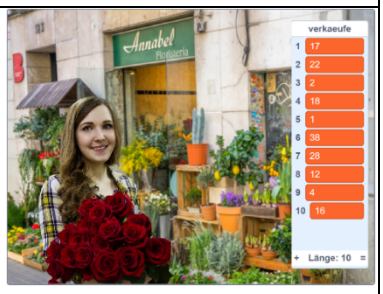
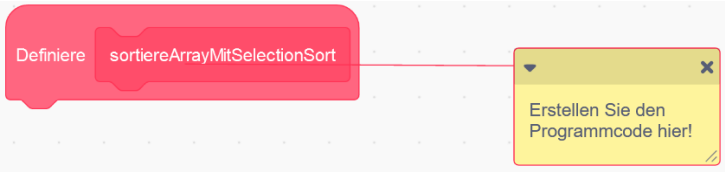
Eva	Anja	Dragan	Jasmin	Alex	
					Durchgang 1
					Durchgang 2
					Durchgang 3
					Durchgang 4

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

3. Öffnen Sie das Projekt *7310_SelectionSort-AUFGABE.* Ihre Aufgabe ist es, Lottozahlen zu sortieren. Setzen Sie dazu das *Selection Sort*-Struktogramm um!



4. Öffnen Sie das Projekt *7320_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 *Selection Sort*-Algorithmus ein!