Ein einfacher Sortieralgorithmus

Leitprogrammartige Unterrichtsunterlagen
von Bruno Lustenberger

Inhalt:
Sortieren ist eine der grundlegenden Aufgaben von Computern und von grosser praktischer Bedeutung. In diesen leitprogrammartigen Unterlagen wird insbesondere der Begriff des Algorithmus sorgfältig eingeführt und entwickelt.

Unterrichtsmethode: Leitprogrammartige Unterrichtsunterlagen
Das Leitprogramm ist ein Selbststudienmaterial. Es enthält alle notwendigen Unterrichtsinhalte, Übungen, Arbeitsanleitungen und Tests, welche die Schüler/innen brauchen, um ohne Lehrperson lernen zu können. Die folgenden Unterlagen nehmen Elemente des Leitprogramms auf.

Fachliches Review:
Juraj Hromkovic, Informationstechnologie und Ausbildung, ETH Zürich

Fachdidaktisches Review:
Juraj Hromkovic, Informationstechnologie und Ausbildung, ETH Zürich

Publiziert auf EducETH:
23. Juni 2008

Ein einfacher Sortieralgorithmus

Fach	Mathematik
Schultyp	Mittelstufe
Alter	Ab 11. Schuljahr
Vorkenntnisse	Variablen im mathematischen Sinne Arbeiten mit Flussdiagrammen Arbeitsweise mit leitprogrammartigen Unterlagen
Bearbeitungsdauer	6 Lektionen
Autor	B. Lustenberger, KS Glattal E-Mail: BMSL@ggaweb.ch
Fassung vom	14. November 2007 / 18. März 2008
Schulerprobung	Noch keine

Inhaltsverzeichnis

Einleitung 2
Sortieren durch Menschen und Maschinen 3
Ein einfacheres Teilproblem: Suchen 5
Definition unserer Maschine 6
Unsere Maschine kann suchen 9
Unsere Maschine kann sortieren 11
Der Suchalgorithmus als Baustein 13
Den Baustein einsetzen 15
Allgemeinheit 16
Korrektheit 17
Zusammenfassung 18
Lernkontrolle 20
Lösungen 21
Quellen 27

Einleitung

Thema

Sortieren ist eine der grundlegenden Aufgaben von Computern und von grosser praktischer Bedeutung. Z. B. muss die Einwohnerkontrolle Personendaten nach Geburtsdatum oder ein Warenhaus seine Artikel nach Preis oder Umsatz sortieren können. Hier spielen Computer eine ihrer Stärken aus: riesige Datenmengen in kurzer Zeit verarbeiten.

Aber wie sortiert ein Computer solche Datenmengen? Er verwendet bestimmte Sortierverfahren. Man nennt das auch: „Sortieralgorithmen“.

Doch was heisst das genau: „Algorithmus“? Darum geht es hier: Welche Eigenschaften muss ein Verfahren haben, damit wir es Algorithmus nennen dürfen?

Die Problemstellung des Sortierens dient uns als Beispiel, anhand dessen wir diese Frage untersuchen. Wir werden ein einfaches Sortierverfahren entwickeln und es schrittweise so präzisieren und verfeinern, dass daraus ein Algorithmus wird.

Ablauf

Sie werden diese Unterlagen selbständig durcharbeiten. Es gibt einige Texte zu lesen und zu studieren, aber vor allem gilt es, Aufgaben zu lösen. Die Aufgaben leiten Sie an, unseren einfachen Sortieralgorithmus weitgehend selbst zu entwickeln. Dabei wird sich auch der Begriff des Algorithmus klären.

Sie werden oft mit Flussdiagrammen arbeiten. Und zwar auf zwei Arten. Einerseits werden Sie Flussdiagramme ausführen, so als wären Sie selbst eine Maschine. Andrerseits, und das ist die Hauptaufgabe, werden Sie selber Flussdiagramme entwerfen, um einer Maschine „klipp und klar“ zu sagen, was sie zu tun hat.

Lösen Sie zuerst jede Aufgabe, bevor Sie mit dem Lesen des Textes weiterfahren. Vergleichen Sie auch Ihre Lösung mit der vorgeschlagenen Lösung im zweitletzten Abschnitt. Das ist hilfreich und manchmal nötig für das Verständnis des folgenden Textes.

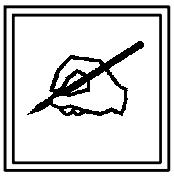
Lernziele

1. Sie kennen die drei Kriterien, die ein Verfahren erfüllen muss, um als Algorithmus zu gelten. Sie merken sich diese mit der Abkürzung **„MAK“.**
2. Sie können einen einfachen Sortieralgorithmus mit einem Flussdiagramm beschreiben.
3. Sie können ein vorgegebenes Flussdiagramm zuverlässig abarbeiten. Sie machen das mit einer Tabelle.
4. Sie können Flussdiagramme einfach und übersichtlich halten, indem Sie Bausteine einsetzen.

Sortieren durch Menschen und Maschinen

Wir Menschen „wissen“, wie man eine Liste von Zahlen sortiert. Wir schauen etwa eine Liste an, verschieben einige Zahlen, dann schauen wir die Liste wieder an, verschieben weitere Zahlen usw. – und schliesslich ist die Liste sortiert.

Doch was machen wir eigentlich? Das ist gar nicht so einfach zu beschreiben.



Aufgabe 1

Hier ist eine Liste von 8 ganzen Zahlen:

23	7	45	19	2	17	11	4

Und darunter eine leere Liste:

Ihre Aufgabe:

1. Sortieren Sie diese Liste, tragen Sie das Resultat in die leere Liste darunter ein. Die Sortierreihenfolge soll aufsteigend sein, d.h. die kleinste Zahl steht in der sortierten Liste ganz links.
2. Überlegen Sie: Wie sind Sie vorgegangen?
3. Beschreiben Sie stichwortartig und falls nützlich mit Bildern, wie Sie vorgegangen sind.
4. Tauschen Sie Ihre schriftliche Beschreibung mit einem Kollegen aus. Versuchen Sie die Beschreibung des Kollegen zu verstehen, ohne mit ihm darüber zu sprechen.

Vermutlich verstehen Sie die Beschreibung Ihres Kollegen. Denn Sie sind eben ein Mensch, Sie können sprachliche und bildliche Beschreibungen verstehen und Sie wissen auch, dass hier eine Liste sortiert werden soll.

Doch nun machen wir ein Gedankenexperiment: Wir wollen unsere Beschreibung nicht einem Menschen sondern einem viel einfacheren Wesen übergeben, einer Maschine. Diese Maschine hat folgende Eigenschaften:

- Sie kann eine begrenzte Anzahl einfacher Tätigkeiten ausführen.
- Sie kann einige einfache Anweisungen, die diese Tätigkeiten beschreiben, lesen und dann die entsprechende Tätigkeit ausführen.
- Sie ist „ahnungslos“. Sie weiss nicht, dass wir eine Liste sortieren wollen. Sie führt nur die Anweisungen aus, die wir ihr geben.

Unser Ziel: Wir wollen eine Beschreibung entwickeln, die aus solchen einfachen Anweisungen aufgebaut ist. Wenn die Maschine nun diese Beschreibung liest und die Anweisungen ausführt, dann wird die Liste sortiert.

(Bemerkung: Natürlich kann eine solche Maschine durch einen Computer realisiert werden. Doch wir müssen uns hier gar nicht mit Computern befassen. Wir können uns z.B. auch einen Menschen vorstellen, der nichts von unserer Absicht des Sortierens weiss und wegen grosser Müdigkeit oder Stress nur noch ganz einfache Anweisungen verstehen und ausführen kann.)

Wir können unsere Zielsetzung auch so formulieren: wir wollen einen Algorithmus für das Sortieren entwickeln. Denn unsere Bezugnahme auf eine einfache Maschine entspricht einem Kriterium, das jeder Algorithmus erfüllen muss.

1. Algorithmus-Kriterium: Maschinentauglichkeit
Ein Algorithmus muss aus so einfachen Tätigkeiten und seine Beschreibung aus so einfachen Anweisungen aufgebaut sein, dass sogar eine einfache, von uns gebaute (oder gedachte) Maschine diese Beschreibung lesen und die Tätigkeiten ausführen kann.

Es gibt noch andere Kriterien, auf diese kommen wir gegen Ende dieses Leitprogramms zu sprechen. Zunächst werden wir jetzt recht viel Aufwand treiben, um dieses erste Kriterium zu erfüllen.

Ein einfacheres Teilproblem: Suchen

Sie haben in Aufgabe 1 eine Liste auf Ihre eigene Weise sortiert und dann Ihr Vorgehen beschrieben. Wahrscheinlich könnte man Ihr Verfahren und dessen Beschreibung so verfeinern, wie es unserem Ziel entspricht.

Nun bitte ich Sie aber, Ihre Aufmerksamkeit einem möglicherweise anderen Verfahren zu schenken. Dieses kann man stichwortartig so beschreiben:

1. **Suche** in der Liste die kleinste Zahl. Vertausche diese mit der Zahl auf dem Platz 1.
2. **Suche** in der Liste ab Platz 2 die kleinste Zahl. Vertausche diese mit der Zahl auf dem Platz 2.
3. Usw.

Lösen Sie die folgende Aufgabe, um sicherzustellen, dass Sie verstehen, wie diese Beschreibung gemeint ist.

Aufgabe 2

Die erste Zeile in der folgenden Tabelle stellt den Anfangszustand der Liste dar. Die folgenden Zeilen sollen den Zustand der Liste nach dem ersten, zweiten, dritten und vierten Schritt des oben beschriebenen Verfahrens darstellen. Füllen Sie diese Zeilen aus.

Anfangszustand:	23	7	45	19	2	17	11	4
Nach 1. Schritt:								
Nach 2. Schritt:								
Nach 3. Schritt:								
Nach 4. Schritt:								

Jetzt sehen Sie sicher, dass die Liste spätestens nach 7 Schritten sortiert ist.

Eine wesentliche Tätigkeit bei diesem Verfahren ist das **Suchen** der kleinsten Zahl in einem Teilabschnitt der Liste. Wir nehmen jetzt an, dass für unsere Maschine diese Tätigkeit bzw. die Anweisung „Suche…“ nicht genügend einfach ist. Wir müssen also das **Suchen in noch einfachere Bestandteile zerlegen**.

Wie weit sollen wir bei dieser Zerlegung in einfachere Bestandteile gehen? „Irgendwo“ müssen wir aufhören. Dieses „Irgendwo“ definieren wir selbst. Die Definition besteht in einer Auflistung der einfachen Tätigkeiten und der sie beschreibenden Anweisungen, die unsere Maschine beherrschen soll.

Definition unserer Maschine

Die einfache Maschine, die wir uns hier vorstellen, kann im Wesentlichen mit Zahlen rechnen und eine Folge von Rechenanweisungen, die in Form eines Flussdiagramms gegeben ist, abarbeiten.

	Tätigkeit	**Anweisung oder Beschreibung**
1	Die Maschine kann die 4 arithmetischen Grundoperationen auf Zahlen anwenden und Zahlen vergleichen.	Die üblichen Operationszeichen: +, –, *, /, =, <, ≤, ≠
2	Die Maschine kann Variablen von Zahlen enthalten.	Variablen werden mit kleinen Buchstaben bezeichnet, z.B. n, k, s
3	Die Maschine kann einer Variable einen konkreten Wert oder den Wert einer anderen Variable zuweisen.	Beispiele: n ← 1 n ← k
4	Der zugewiesene Wert kann auch durch einen Ausdruck, der Variablen enthält, berechnet werden. Der Ausdruck kann sogar die gleiche Variable enthalten.	n ← k + 1 n ← k + s n ← n + 1
5	Die Maschine kann auch Listen von ganzen Zahlen enthalten. Dabei verhält sich jeder Platz der Liste wie eine Variable.	Eine Liste wird mit einem grossen Buchstaben bezeichnet, z.B: L. L[1]: die erste Zahl in der Liste. L[2]: die zweite Zahl in der Liste. Usw.
6	Um auf die Zahl an einem bestimmten Platz der Liste zuzugreifen, kann auch eine Variable eingesetzt werden.	Beispiel mit Variable k: L[k]: die Zahl am k-ten Platz der Liste.
7	Die Maschine kann eine Folge von Anweisungen, die in der Form eines Flussdiagramms gegeben ist, abarbeiten.	Die folgenden Flussdiagramm-Elemente: Start, Stop, Anweisungsbox, Entscheidungsbox, Pfeile.

Die folgenden zwei Aufgaben machen Sie mit unserer kleinen Maschine vertraut. Die Liste L ist die von unserem Beispiel aus Aufgabe 1.

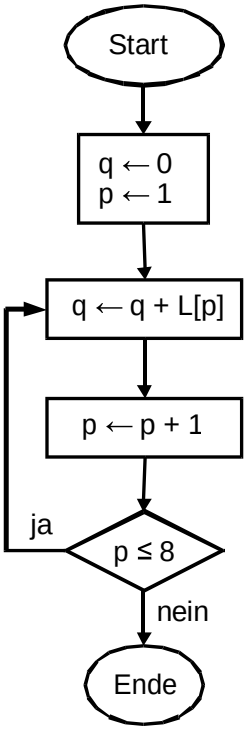
Aufgabe 3

Im folgenden sehen Sie links ein Flussdiagramm und rechts eine Tabelle mit den Variablen, die in diesem Flussdiagramm vorkommen.

1. Führen Sie nun das Flussdiagramm so aus, als wären sie selbst die Maschine. Füllen Sie dazu in der Tabelle die Werte ein, welche die Variablen während der Ausführung annehmen. Für die ersten 2 Zeilen ist das schon gemacht.
2. Wenn Sie das Flussdiagramm fertig abgearbeitet haben, enthält die Variable q einen bestimmten Wert. Was stellt dieser Wert dar?

Unsere Liste L war die folgende:

23	7	45	19	2	17	11	4



q	**p**
0	1
23	2

Abarbeitungstabelle
Eine Tabelle, wie Sie sie soeben ausgeführt haben, nennen wir Abarbeitungstabelle. Dies ist ein nützliches Hilfsmittel, um ein Flussdiagramm zuverlässig auszuführen.

Aufgabe 4

Schreiben Sie ein Flussdiagramm mit Anweisungen, so dass in der Liste L alle Zahlen, die grösser als 10 sind, auf 0 gesetzt werden.
(Natürlich sollen Sie das Flussdiagramm so schreiben, dass es auch funktioniert, wenn L andere Zahlen als die von Aufgabe 1 enthält.)

Machen Sie eine Abarbeitungstabelle, die zeigt, wie sich die Liste von Aufgabe 1 beim Durchlaufen Ihres Flussdiagramms verändert.

Unsere Maschine kann suchen

Jetzt sind wir in der Lage, unserer Maschine die nötigen Anweisungen zu geben, so dass sie für uns „die kleinste Zahl in der Liste L“ findet. Doch Achtung: Wir wollen das Suchen später für das Sortieren verwenden. Und hier interessiert uns nicht etwa der Wert der kleinsten Zahl, sondern der Platz innerhalb der Liste, an dem sich diese Zahl befindet!

Wir verwenden die Werte der Aufgabe 1:

23	7	45	19	2	17	11	4

Uns interessiert nicht der Wert 2, sondern die „Platznummer“ 5, an welcher sich die Zahl 2 befindet.

Index
Die Platznummer innerhalb einer Liste nennen wir auch Index.

Jetzt können wir auch sagen: **Unsere Maschine soll den Index der kleinsten Zahl in der Liste L finden**. Natürlich soll sie dies auch leisten, wenn die Liste L andere Werte als die von Aufgabe 1 enthält.

Wie übergibt die Maschine diesen Index an uns? Das müssen wir genau festlegen, wenn wir das Suchen später für das Sortieren verwenden wollen. Am einfachsten ist es, wenn die Maschine den Index in einer bestimmten Variable ablegt. Wir legen hier fest: **Unsere Maschine soll den gefundenen Index in der Variable s ablegen**.

Hier ist eine Beschreibung eines möglichen Suchverfahrens. Die Idee ist: Wir gehen von links nach rechts der Liste entlang. Wenn immer wir eine kleinere Zahl finden, merken wir uns deren Index als vorläufiges Resultat. Wenn wir am Schluss der Liste angekommen sind, ist das Resultat endgültig.

Diese Beschreibung ist natürlich noch viel zu kompliziert für unsere Maschine. Wir müssen ihr dieses Verfahren mit Hilfe von Variablen und Operationen beschreiben.

Für das vorläufige Resultat verwenden wir die Variable s. Für das Entlanggehen verwenden wir eine Hilfsvariable r, welche über die Indexwerte 1,2,3… usw. läuft.

Jetzt können wir das Verfahren genauer beschreiben:

1. Da die kleinste Zahl auch ganz am Anfang stehen könnte, setzen wir zuerst einmal s auf 1. Das ist unser erstes vorläufiges Resultat. Da wir als nächstes die Zahl an der zweiten Stelle mit der an der ersten Stelle vergleichen müssen, setzen wir die Hilfsvariable r auf 2.
2. Jetzt vergleichen wir die Zahl L[s] mit der Zahl L[r]. Falls L[r] kleiner ist, so haben wir eine neue kleinere Zahl gefunden und setzen deshalb s auf r; sonst bleibt s unverändert.
3. Nun gehen wir um 1 weiter entlang der Liste, d.h. wir erhöhen r um 1.
4. Hier fahren wir wieder bei Schritt 2 weiter, ausser wir haben schon die ganze Liste abgesucht.

Auch diese Beschreibung ist für unsere Maschine noch nicht tauglich. Doch Sie sind ja inzwischen ein Spezialist für unsere Maschine und können eine taugliche Beschreibung entwickeln!

Aufgabe 5

Übersetzen Sie die obige Beschreibung in ein Flussdiagramm für unsere Maschine. Achten Sie besonders darauf, dass Ihr Verfahren auch dann funktioniert, wenn die kleinste Zahl ganz am Anfang oder am Schluss der Liste steht.
„Testen“ Sie deshalb Ihr Verfahren, indem Sie es für folgende 3 Listen ausführen:

23	7	45	19	2	17	11	4

2	17	45	19	23	7	11	4

23	7	17	11	4	45	19	2

Testen heisst hier: Sie machen für jede der 3 Listen eine Abarbeitungstabelle.

Nachdem Sie diese Aufgabe gelöst haben, sind Sie sicher einverstanden mit der Aussage, dass unsere Maschine suchen kann.

Beachten Sie: Im Folgenden werden wir das Suchverfahren, das im Lösungsabschnitt beschrieben ist, weiterverwenden. Ihr Suchverfahren sieht vermutlich nicht genau gleich aus.

Unsere Maschine kann sortieren

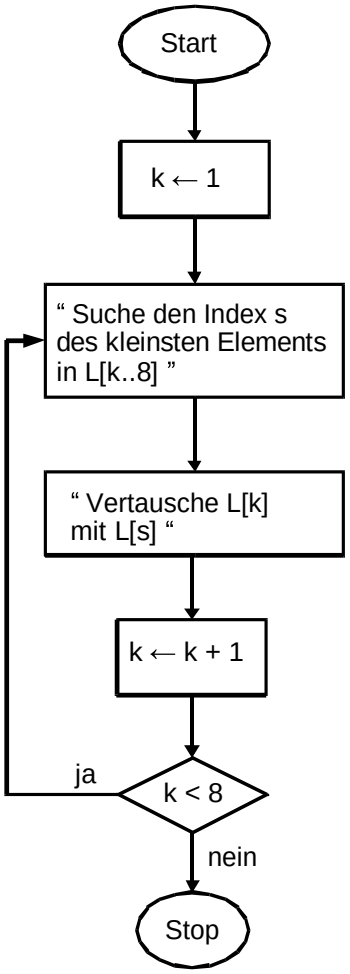
Mit dem Suchen haben wir eigentlich den schwierigsten Teil unseres Sortierverfahrens schon gemeistert. Wir können dies als Teilverfahren einbauen und damit wird unser Sortierverfahren recht einfach.

Rechts sehen Sie das Flussdiagramm, das unserer Beschreibung auf Seite 5 entspricht.

Das Flussdiagramm enthält zwei Anweisungsboxen, die noch nicht maschinentauglich sind. Der Text ist deshalb in Anführungszeichen gesetzt.

Die Schreibweise L[k .. 8] bedeutet: die Liste vom k-ten bis zum 8-ten Platz.

Als nächstes wollen wir diese Anweisungsboxen so verfeinern, dass unsere Maschine sie problemlos ausführen kann.



Kümmern wir uns zuerst um die einfachere Anweisungsbox, das Vertauschen. Die Werte von zwei Variablen zu vertauschen, ist eine einfache Tätigkeit. Wir dürfen annehmen, dass unsere Maschine dies beherrscht. Deswegen fügen wir das einfach zu den Fähigkeiten unserer Maschine hinzu.

	Tätigkeit	**Anweisung oder Beschreibung**
8	Die Maschine kann die Werte von zwei Variablen vertauschen. Die Variablen können auch Plätze in einer Liste sein.	u ↔ k L[5] ↔ L[3]

Aufgabe 6

Schreiben Sie ein Flussdiagramm, das die Reihenfolge der Zahlen in unserer Liste umkehrt. Das heisst:

aus	23	7	45	19	2	17	11	4
wird	4	11	17	2	19	45	7	23

Jetzt zur Teilaufgabe des Suchens. Zunächst müssen wir unser Suchverfahren etwas flexibler machen. Wir suchen nämlich das kleinste Element nicht nur in der ganzen Liste, sondern auch bloss in einem Teil der Liste, z. B. im Teil vom 3-ten bis zum 8-ten Platz. Diesen Teilabschnitt bezeichnen wir so: L[3..8].

Anschaulich: L[3..8]

23	7	45	19	2	17	11	4

Die untere Grenze des Listenabschnitts, in dem wir suchen, muss also flexibel sein.

Und wir gehen gleich noch einen Schritt weiter: Wir wollen nicht nur die untere, sondern auch die obere Grenze des Listenabschnitts flexibel machen. Wir brauchen das zwar im Moment nicht, aber es ist klar, dass diese Flexibilität für eine breite Verwendbarkeit unseres Suchverfahrens vorteilhaft ist.

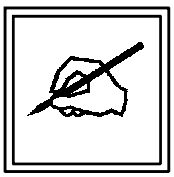
Beispiel für L[3..7]:

23	7	45	19	2	17	11	4
		↑ 3				↑ 7	

Wenn der Index der unteren Grenze in der Variable u und der Index der oberen Grenze in der Variable v enthalten ist, so bezeichnen wir den entsprechenden Listenabschnitt mit L[u..v]. Wenn u=3 und v=7 sind, können wir also auch schreiben:

Beispiel für L[u..v]:

23	7	45	19	2	17	11	4
		↑ u				↑ v	



Aufgabe 7

Ändern Sie unser Suchverfahren so ab, dass der Index des kleinsten Elements in einem Teilabschnitt L[u..v] gesucht wird. Die Werte der Variablen u und v sind vorgegeben.

Für das Suchverfahren darf man annehmen, dass u < v und dass keiner der Indizes ausserhalb der Liste liegt (z.B. u=0 oder v=9 kommt nicht vor)

Der Suchalgorithmus als Baustein

Nun haben wir ein flexibel einsetzbares Suchverfahren. Wie können wir es in unser Flussdiagramm für das Sortieren **einbauen**?

Eine erste Möglichkeit ist, die ganze Anweisungsbox „Suche...“ zu ersetzen durch das Flussdiagramm für das Suchverfahren. Start und Ende ersetzen wir durch die ein- und ausgehenden Pfeile. Zusätzlich müssen wir vorher noch eine Anweisungsbox einführen, welche den Variablen u und v die aktuellen Werte zuweist:
u ← k
v ← 8

Doch diese Möglichkeit wollen wir hier nicht weiterverfolgen, und zwar aus folgenden Gründen:

1. Das Flussdiagramm wird so ziemlich gross und unübersichtlich.
2. Unsere Anweisungsbox „Suche...“ ist eigentlich eine Art **Baustein**. Wir, die menschlichen Leser des Flussdiagramms, sehen sofort, dass unsere Maschine hier eine **zusammengesetzte Tätigkeit** ausführen soll, die einem ganz bestimmten Zweck (eben dem Suchen) dient. Diese im Flussdiagramm vorhandene Information möchten wir nicht zerstören.

Die andere Möglichkeit ist deshalb, den Baustein „Suche...“ beizubehalten. Dazu erweitern wir noch einmal die Fähigkeiten und erlaubten Anweisungen unserer Maschine. Die neue Anweisung muss so präzis sein, dass die Maschine sie ohne „Missverständnisse“ ausführen kann.

<table>
<tr><th></th><th>Tätigkeit</th><th>Anweisung oder Beschreibung</th></tr>
<tr><td>9</td><td>Die Maschine kann in einem Listenabschnitt den Index der kleinsten Zahl finden.</td><td>IndexOfSmallest(L,u,v;s;r)
<table>
<tr><td>L:</td><td>Die Liste</td></tr>
<tr><td>u,v:</td><td>Untere und obere Grenze des Listenabschnitts. Annahmen:
1) u < v
2) u und v sind innerhalb des Indexbereichs der Liste L.</td></tr>
<tr><td>s:</td><td>In dieser Variable wird der gefundene Index abgelegt.</td></tr>
<tr><td>r:</td><td>Eine interne Arbeitsvariable.</td></tr>
</table></td></tr>
</table>

Makro
Einen Baustein dieser Art nennt man auch **Makro**. Die Variablen L,u,v usw. heissen **Parameter** des Makros. Wird ein solcher Baustein in einem Flussdiagramm eingesetzt, so spricht man auch von einem **Aufruf** des Makros.

Wieso verwendet man überhaupt Parameter? Unser Makro könnte ja einfach fix mit den Variablen L, u,v,s,r arbeiten! Doch dies wäre zu wenig flexibel. Z. B. wollen wir unser Makro auch in einem anderen Flussdiagramm, in dem die Variablen u und v schon für etwas anderes gebraucht werden, einsetzen können. Wenn nun z.B. die Variablen f und g frei sind, können wir diese für die Grenzen des Listenabschnitts verwenden und unser Makro so aufrufen:

IndexOfSmallest(L,f,g;s;r)

Jetzt wird genau das gleiche Flussdiagramm ausgeführt, nur wird anstatt der Variablen u die Variable f und anstatt der Variablen v die Variable g verwendet.

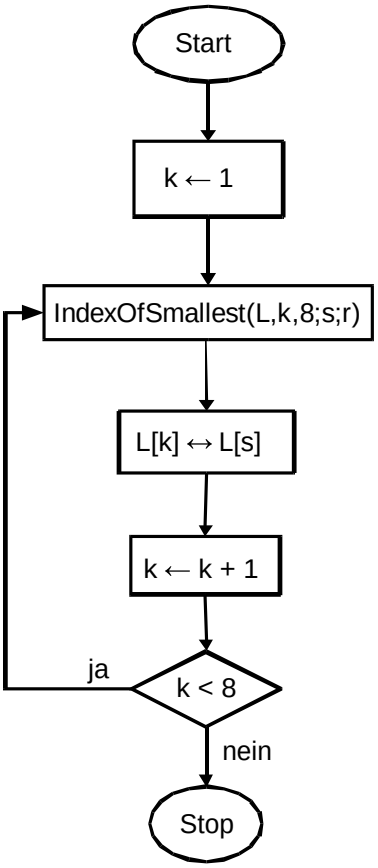
Ferner können wir anstatt u oder v auch fixe Zahlen einsetzen, z.B.:

IndexOfSmallest(L,u,8;s;r)

Wieder wird das gleiche Flussdiagramm ausgeführt, es ist bloss überall v durch 8 ersetzt.

Den Baustein einsetzen

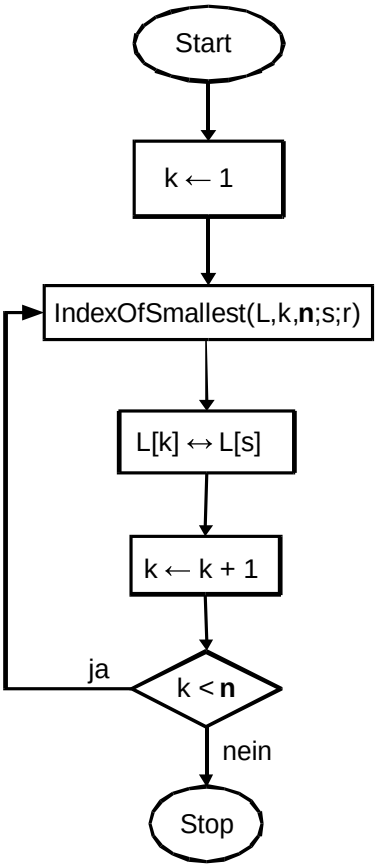
Mit den beiden Erweiterungen unserer Maschine haben wir alles beisammen, um das Suchverfahren auf maschinentaugliche Weise zu beschreiben. Wir können die beiden Anweisungsboxen, deren Text in Anführungszeichen gesetzt ist, durch maschinentaugliche Anweisungen ersetzen:



Allgemeinheit

Natürlich sind wir noch nicht zufrieden. Denn bisher haben wir ausschliesslich Listen der Länge 8 behandelt. Selbstverständlich wollen wir Listen beliebiger Länge sortieren können.

Wir nehmen jetzt an, dass die Länge der Liste durch eine Variable n gegeben ist. Die Verallgemeinerung folgt sofort: wir ersetzen im Flussdiagramm einfach 8 durch n.



Hier liegt allerdings noch ein mögliches Problem verborgen. Doch darum kümmern wir uns im nächsten Abschnitt. Zunächst halten wir fest:

2. Algorithmus-Kriterium: Allgemeinheit
Ein Algorithmus muss **auf alle Problemfälle** eines gegebenen Problems **anwendbar** sein. Z. B. muss ein Sortieralgorithmus auf Listen von beliebigen Zahlen und beliebiger Länge anwendbar sein.

Aufgabe 8

Nehmen Sie das Flussdiagramm von Aufgabe 6. Ändern Sie es so ab, dass für eine Liste beliebiger Länge n ≥ 1 die Reihenfolge umgekehrt wird.

Korrektheit

Lösen Sie die folgende Aufgabe, bevor Sie im Text weiter lesen!

Aufgabe 9

Untersuchen Sie unseren Suchalgorithmus für den Fall, dass die Liste die Länge n = 1 hat. Was passiert?

Natürlich kann man einwenden, dass es nicht sinnvoll ist, eine Liste der Länge 1 zu sortieren. Wenn unser Suchalgorithmus aber innerhalb einer umfassenderen Aufgabe als Makro aufgerufen wird, kann es durchaus sein, dass einmal die Länge der zu sortierenden Liste 1 (oder sogar 0 oder negativ) ist.

Natürlich könnten wir definieren, dass unser Suchalgorithmus die Annahme:

$n > 1$

macht, und damit die „Verantwortung“ auf die Umgebung abschieben. (Etwas Analoges haben wir beim Suchalgorithmus gemacht). Doch es ist vorsichtiger, möglichst wenige oder keine Annahmen zu machen. Wir verlangen deshalb von unserem Sortieralgorithmus, dass er sich auch bei $n \leq 1$ korrekt verhält.

Aufgabe 10

Ändern Sie das Flussdiagramm unseres Sortieralgorithmus so ab, dass er sich auch bei $n \leq 1$ korrekt verhält.

Wir halten zunächst fest:

3. Algorithmus-Kriterium: Korrektheit
Ein Algorithmus muss **für alle Problemfälle** eines gegebenen Problems die **korrekte Lösung** liefern. Das bedeutet insbesondere auch, dass der Algorithmus für jeden Problemfall seine Arbeit **in endlicher Zeit abschliesst**.

Die Forderung nach Abschluss in endlicher Zeit kommt Ihnen vielleicht seltsam vor. Sie finden in der Lernkontrolle eine Aufgabe dazu.

Zusammenfassung

Schauen wir zurück! Im Wesentlichen sind wir zwei Fragen nachgegangen:

- Was ist ein Algorithmus?
- Wie beschreibt man einen Algorithmus?

Die Fragen sind miteinander verknüpft. Trotzdem ist es richtig, sie auseinander-zuhalten.

Zunächst zur ersten Frage. Wir haben 3 Kriterien formuliert, die ein Verfahren erfüllen muss, um die Bezeichnung Algorithmus zu verdienen. Man kann sich diese Kriterien gut mit der Abkürzung **MAK** und der folgenden Tabelle merken:

M	**Maschinentauglich**	Ein Algorithmus kann von einer einfachen Maschine ausgeführt werden.
A	**Allgemein**	Ein Algorithmus ist auf alle Fälle eines gegebenen Problems anwendbar.
K	**Korrekt**	Ein Algorithmus liefert für alle Fälle eines gegebenen Problems in endlicher Zeit die richtige Lösung.

Jetzt zur zweiten Frage. Die Antwort ergibt sich eigentlich aus dem Kriterium der Maschinentauglichkeit: **Die Beschreibung muss derart sein, dass eine Maschine diese Beschreibung lesen und die beschriebenen einfachen Tätigkeiten ausführen kann**.

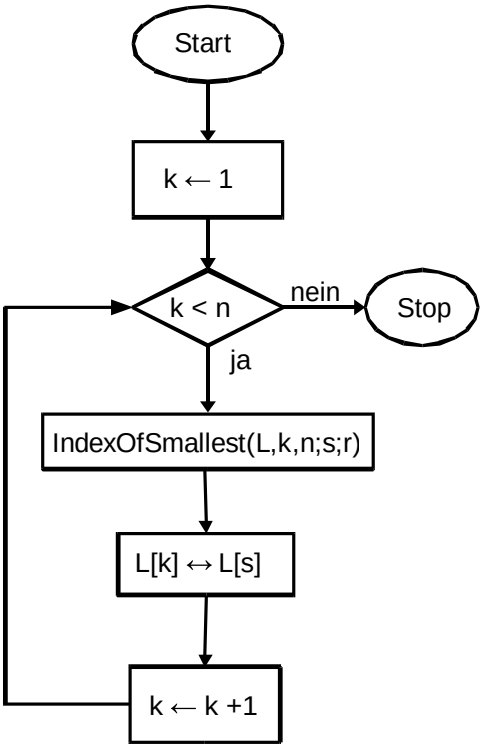
Es gibt viele verschiedene Arten, Algorithmen zu beschreiben. Meistens geschieht das in Form von sogenannten Programmiersprachen. Das ist aber nicht zwingend. Wir haben uns eine einfache Maschine ausgedacht, ihre möglichen Tätigkeiten und die zugehörigen Anweisungen definiert und dann Flussdiagramme verwendet.

Das Beispiel, an dem wir unsere Überlegungen entwickelt haben, war das Sortieren einer Liste von Zahlen. Sie kennen jetzt einen möglichen Sortieralgorithmus. Dieser heisst übrigens **Selectionsort**. Es gibt noch andere, raffiniertere Sortieralgorithmen. Diese werden in der Praxis häufiger eingesetzt, weil sie schneller sind.

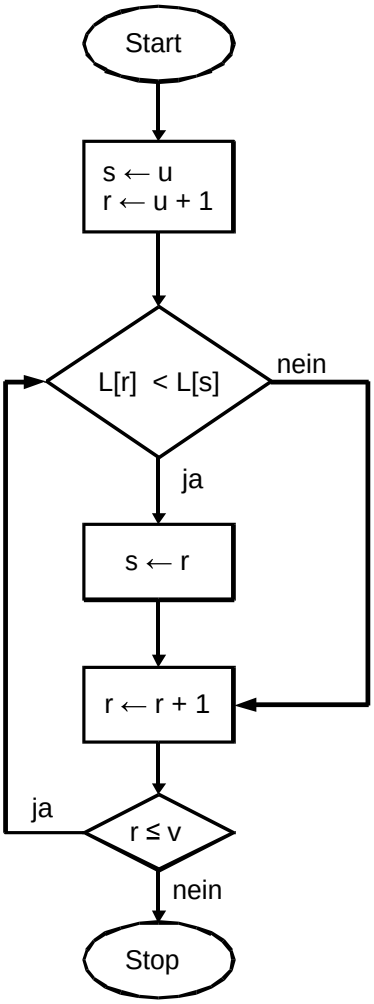
Ein wesentlicher Teil unseres Suchalgorithmus besteht in der Teilaufgabe des Suchens. Wir haben dafür ein separates Flussdiagramm entwickelt und dieses als Baustein in das übergeordnete Flussdiagramm des Sortierens eingebaut. Damit bleibt Übersichtlichkeit und Verständlichkeit erhalten. Solche Bausteine nennen wir **Makros**. Der Grundsatz hinter einer solchen Beschreibungsweise heisst **Modularisierung**.

Die beiden definitiven Flussdiagramme zum Sortieren und Suchen finden Sie auf der folgenden Seite.

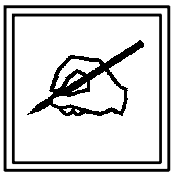
Sortiere die Liste L der Länge n:



IndexOfSmallest(L,u,v;s;r):



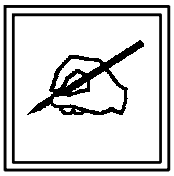
Lernkontrolle



Aufgabe 11

In unseren Beispielen von zu sortierenden Listen waren alle Zahlen voneinander verschieden. Insbesondere gab es genau eine kleinste Zahl in der Liste.

Was liefert unser Suchalgorithmus, wenn in der Liste bzw. im abzusuchenden Listenabschnitt die kleinste Zahl mehrfach vorkommt?



Aufgabe 12

Entwickeln Sie für unsere Maschine einen Algorithmus, der zählt wie oft die kleinste Zahl in der Liste vorkommt.

Sie dürfen (müssen aber nicht) das Makro IndexOfSmallest verwenden.

Aufgabe 13

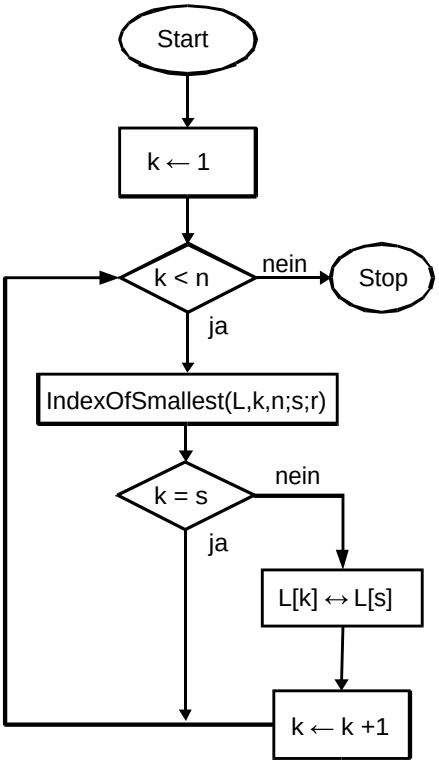
Bei unserem Sortieralgorithmus kann etwas Seltsames passieren. Der vom Makro IndexOfSmallest gefundene Index s kann gleich k sein. Dann wird in der nächsten Anweisung also L[k] mit sich selber vertauscht.

Das ist zwar nicht falsch, aber überflüssig. Man könnte deshalb auf die Idee kommen, in diesem Fall die Vertauschung einzusparen und den Sortieralgorithmus so wie im nebenstehenden Flussdiagramm zu verbessern.

Doch in dieser Verbesserung steckt ein Fehler.

- Finden Sie diesen Fehler!
- Welches Algorithmus-Kriterium ist hier verletzt?
- Korrigieren Sie das Flussdiagramm.

Hinweis: Der Fehler tritt z.B. auf, wenn die Liste von Aufgabe 1 sortiert werden soll.



Lösungen

Aufgabe 1

-

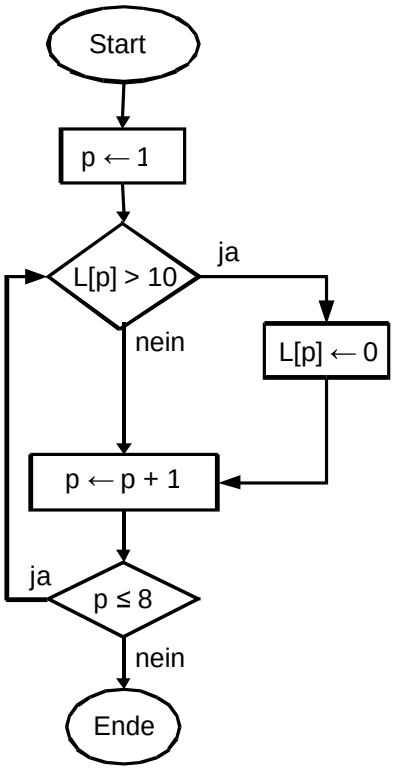
Aufgabe 2

Anfangszustand:	23	7	45	19	2	17	11	4
Nach 1. Schritt:	2	7	45	19	23	17	11	4
Nach 2. Schritt:	2	4	45	19	23	17	11	7
Nach 3. Schritt:	2	4	7	19	23	17	11	45
Nach 4. Schritt:	2	4	7	11	23	17	19	45

Aufgabe 3

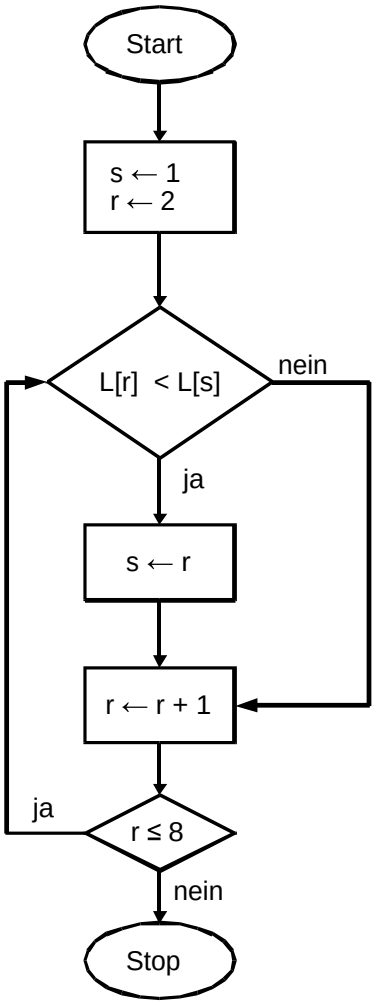
q	**p**
0	1
23	2
30	3
75	4
94	5
96	6
113	7
124	8
128	9

Aufgabe 4



p	**Liste L**							
-	23	7	45	19	2	17	11	4
1	0	7	45	19	2	17	11	4
2	0	7	45	19	2	17	11	4
3	0	7	0	19	2	17	11	4
4	0	7	0	0	2	17	11	4
5	0	7	0	0	2	17	11	4
6	0	7	0	0	2	0	11	4
7	0	7	0	0	2	0	0	4
8	0	7	0	0	2	0	0	4
9	0	7	0	0	2	0	0	4

Aufgabe 5



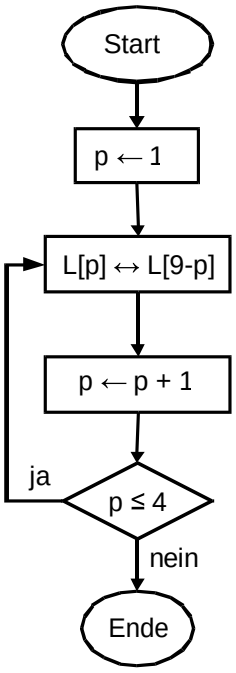
Hier die 3 Abarbeitungstabellen. Die linke gehört zur obersten, die rechte zur untersten Liste.

r	**s**
2	1
3	2
4	2
5	2
6	5
7	5
8	5
9	5

r	**s**
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1

r	**s**
2	1
3	2
4	2
5	2
6	5
7	5
8	5
9	8

Aufgabe 6

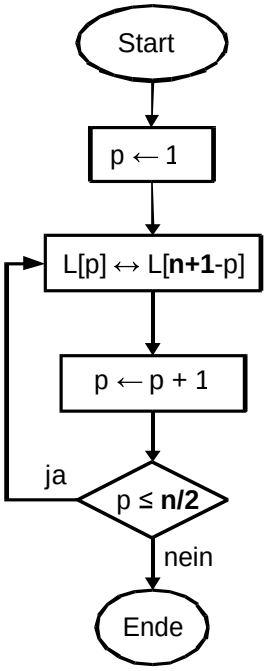


p	**Liste L**							
-	23	7	45	19	2	17	11	4
1	4	7	45	19	2	17	11	23
2	4	11	45	19	2	17	7	23
3	4	11	17	19	2	45	7	23
4	4	11	17	2	19	45	7	23
5	4	11	17	2	19	45	7	23

Aufgabe 7

Siehe das Diagramm rechts im Abschnitt „Zusammenfassung“, S. 19.

Aufgabe 8



Aufgabe 9

Im Falle n=1 wird das Makro IndexOfSmallest mit ungültigen Parameterwerten aufgerufen, nämlich u=1 und v=1. Das führt dazu, dass auf das Listenelement „L[2]“ zugegriffen wird. Doch dieses ist gar nicht definiert, da die Liste ja die Länge 1 hat!

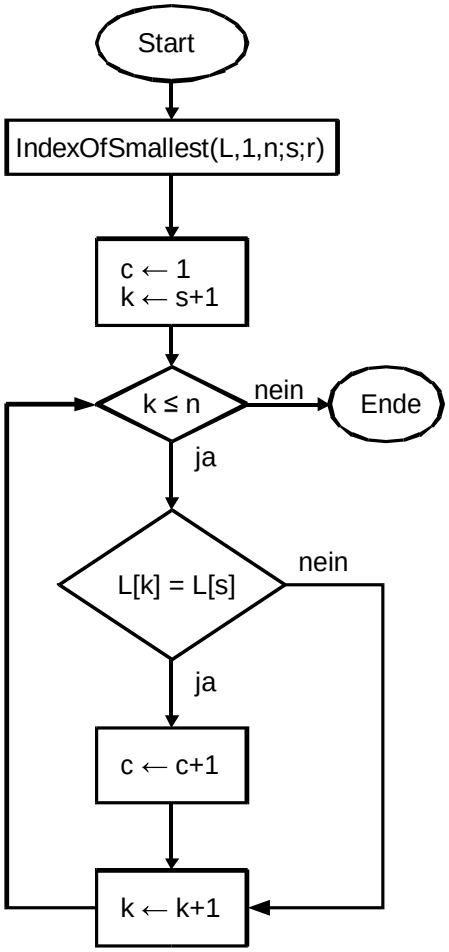
Aufgabe 10

Siehe das Diagramm links im Abschnitt „Zusammenfassung“, S. 19

Aufgabe 11

Der Suchalgorithmus liefert den Index der ersten kleinsten Zahl.

Aufgabe 12

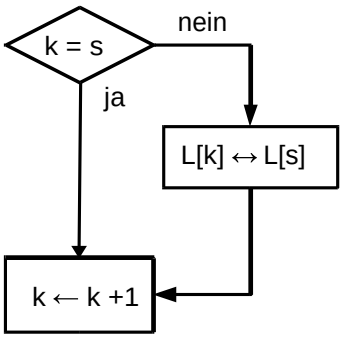


Die Anzahl kleinste Elemente wird in der Variable c abgelegt. Der Algorithmus benützt, dass IndexOfSmallest den Index des ersten kleinsten Elements liefert.

Aufgabe 13

Im Falle k = s wird die Variable k nicht mehr erhöht. Das führt dazu, dass der Algorithmus in diesem Fall seine Arbeit gar nie abschliesst. Das ist also ein Beispiel für eine Verletzung der Anforderung „Abschluss in endlicher Zeit“, die im dritten Algorithmus-Kriterium enthalten ist.

Korrektur: die Anweisung für die Erhöhung von k muss in jedem Fall durchlaufen werden. Der untere Teil des Flussdiagramms muss so aussehen:



Quellen

A) Als Grundlage benützte Quellen

Nr.	Link, Publikation ▪ **Wozu habe ich dies benützt**
1	Jurai Hromkovič: Sieben Wunder der Informatik. Wiesbaden (Teubner). Oktober 2006. 1. Auflage. ▪ Die 3 Kriterien für einen Algorithmus, Seite 62-63.

B) Zitierte Quellen

Keine.