Kapitel 3

Algorithmen

Unter einem Algorithmus versteht man eine genau definierte Handlungsvorschrift zur Lösung eines Problems oder eines bestimmten Typs von Problemen. Eine exakte Definition dieses Begriffes ist nicht trivial und erfolgt in Informatikvorlesungen. Wir werden Algorithmen zur Lösung mathematischer Aufgabenstellungen verwenden.

Beispiel 3.1 *Quadratische Gleichung.* Zur Lösung der quadratischen Gleichung

$$0 = x^2 + px + q, \quad p, q \in \mathbb{R},$$

im Bereich der reellen Zahlen kann man folgenden Algorithmus verwenden:

Algorithmus 3.2 Lösung der quadratischen Gleichung.

- berechne
$$a := -\frac{p}{2}$$
- berechne
$$D := a^2 - q$$
- falls $D \geq 0$, dann
$$x_1 := a + \sqrt{D}, \quad x_2 := a - \sqrt{D}$$
sonst Ausgabe *keine reelle Lösung*

Man kann aber auch einen anderen Algorithmus verwenden, der sich im letzten Schritt unterscheidet und die Vieta[1]sche Wurzelformel nutzt:

Algorithmus 3.3 Lösung der quadratischen Gleichung.

- berechne
$$a := -\frac{p}{2}$$
- berechne
$$D := a^2 - q$$
- falls $D < 0$, dann Ausgabe *keine reelle Lösung*
sonst
  - falls $a < 0$, setze
$$x_1 := a - \sqrt{D}, \quad x_2 := q/x_1$$
sonst falls $a > 0$, setze
$$x_1 := a + \sqrt{D}, \quad x_2 := q/x_1$$
sonst
$$x_1 := \sqrt{-q}, \quad x_2 := -x_1.$$

[1]Francois Vieta (Viète) 1540 – 1603

Beide Algorithmen liefern mathematisch dasselbe Ergebnis. In den Übungen wird man sehen, dass das auf einem Computer im allgemeinen nicht mehr der Fall ist.

Diese Algorithmen enthalten bereits ein wichtiges Merkmal, nämlich Verzweigungen. Man muss an hand von berechneten Werten sich zwischen zwei oder mehr Möglichkeiten entscheiden, welcher Teil des Algorithmus abzuarbeiten ist. □

Beispiel 3.4 *Summe von Zahlen.* Man solle die Summe der natürlichen Zahlen von 1000 bis 10000 bilden. Ein möglicher Algorithmus ist, diese nacheinander aufzuaddieren:

Algorithmus 3.5

- setze $summe := 1000$
- setze für $i = 1001$ bis $i = 10000$

$$summe := summe + i$$

Hierbei ist *summe* als Variable zu verstehen, die einen bestimmten Speicherplatz im Computer einnimmt. Der neue Wert wird somit auf demselben Platz gespeichert, wie der bisherige Wert. Man muss zur Berechnung der Summe 9000 Additionen durchführen.

Auch dieser Algorithmus hat ein charakteristisches Merkmal, nämlich eine Schleife. Es gibt unterschiedliche Arten von Schleifen, siehe später.

Ein anderer Algorithmus nutzt die bekannte Summationsformel von Gauß[2]

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Damit erhält man

$$\sum_{i=1000}^{10000} i = \sum_{i=1}^{10000} i - \sum_{i=1}^{999} i = \frac{10000 \cdot 10001}{2} - \frac{999 \cdot 1000}{2} = 5000 \cdot 10001 - 500 \cdot 999.$$

Damit kann man das Ergebnis mit zwei Multiplikationen und einer Subtraktion berechnen.

Man sieht, es kann billige (effiziente) und teure Algorithmen geben, die zum gleichen Ziel führen. Ziel ist es natürlich, den jeweils effizientesten Algorithmus zu verwenden. □

Zur Beschreibung von Algorithmen nutzt man die folgenden Grundstrukturen:
- Anweisung,
- bedingte Verzweigung (Alternative),
- Wiederholung (Schleife, Zyklus).

Die Zusammenfassung mehrerer Anweisungen wird auch Sequenz genannt.

Beispiel 3.6 Am Anfang von Algorithmus 3.2 kommt die folgende Sequenz

```
a = -p/2;
D = a*a-q;
```

(Hier wird MATLAB–Notation genutzt.) □

Bei der Alternative unterscheidet man die einfache und die vollständige Alternative.

Beispiel 3.7 Die einfache Alternative haben wir bereits in den Algorithmen 3.2 und 3.3 gesehen:

[2] Johann Carl Friedrich Gauss (1777 – 1855)

```
if D<0
   disp('Keine reelle Loesung')
else
   Anweisungen zur Berechnung der Lösung
end
```

Im Algorithmus 3.3 findet man auch eine vollständige Alternative:

```
if a<0
   x_1 = a-sqrt(D);
   x_2 = q/x_1;
elseif a>0
   x_1 = a+sqrt(D);
   x_2 = q/x_1;
else
   x_1 = sqrt(-q);
   x_2 = -x_1;
end
```

Möglichkeiten zur einfachen Behandlung mehrerer Alternativen sind in MATLAB und C ebenfalls vorhanden: `switch`–Anweisung, siehe später. □

In einer Schleife oder einem Zyklus ist eine Sequenz wiederholt abzuarbeiten. Die Anzahl der Durchläufe muss dabei vorab nicht unbedingt bekannt sein, sie bestimmt sich oft im Laufe der Abarbeitung. Man unterscheidet abweisende (kopfgesteuerte) Zyklen und nichtabweisende (fußgesteuerte) Zyklen:

- abweisender Zyklus: Bedingung für die Abarbeitung wird vor Beginn des Zyklus getestet, es kann passieren, dass diese beim ersten Mal schon nicht erfüllt ist und der Zyklus wird nie abgearbeitet,
- nichtabweisender Zyklus: Bedingung für die Abarbeitung wird am Ende des Zyklus getestet, damit wird der Zyklus mindestens einmal abgearbeitet.

Beispiele dafür wird es in den Übungen geben. Ist die Anzahl der Durchläufe bekannt, dann wird der Zyklus durch einen Zähler gesteuert.

Beispiel 3.8 Die Steuerung durch einen Zähler hatten wir bereits in Algorithmus 3.5:

```
for i=1001:10000
   summe = summe + i;
end
```

□

Jeder Zyklus erfordert Vorbereitungen, das heißt, vor Eintritt in den Zyklus sind entsprechende Variablen zu belegen. Im obigen Beispiel ist `summe` = `1000` zu setzen. Im Zykluskörper selbst muss so auf die Abbruchbedingung eingewirkt werden, dass der Abbruch nach endlich vielen Durchläufen garantiert ist. Hat man dabei einen Fehler gemacht, dann kann das Programm den Zyklus nicht verlassen und es hilft nur, das Programm abzubrechen und den Fehler zu suchen.