I. Algorithmen

Computer dienen dazu, ihnen gestellte Aufgaben selbständig zu lösen. Dazu muß ihnen gesagt werden, **wie** sie dies tun sollen. Solch eine Beschreibung heißt Algorithmus. Ein Algorithmus ist eine Folge von Anweisungen, die ein System, das nur diskrete Zustände einnehmen kann, von einem Anfangszustand in einen Endzustand transformiert. Jede Anweisung ruft einen Zustandsübergang hervor, d.h. ein Algorithmus ist ein Verfahren zur schrittweisen Lösung eines Problems.

Die Abarbeitung der Schritte heißt Prozess, daher der Name Prozessor. Ein Computer ist also nur ein ganz spezieller Prozessor. Damit ein Algorithmus von einem Prozessor abgearbeitet werden kann, muß dieser ihn verstehen und jeden einzelnen Schritt ausführen können.

Für die Beschreibung von Algorithmen kann man verschiedene Formen wählen. Die Beschreibung kann zum Beispiel erfolgen in

- Umgangssprache (Gefahr von Mißverständnissen)
- Struktogramm / Flußdiagramm
- Programmiersprache
- mathematischem Formalismus

Nicht immer existiert für ein gegebenes Problem auch eine Algorithmus. Zum Beispiel gibt es keinen Algorithmus, der für ein beliebiges Programm mit beliebigen Eingabedaten die Frage beantwortet, ob das Programm mit diesen Eingabedaten terminiert. Solche Probleme heißen nicht berechenbar. Daneben gibt es Probleme, deren Lösungsalgorithmus zwar existiert, bei dem aber die Komplexität so hoch ist, dass die Lösung zu viel Zeit beansprucht. Darauf beruhen zum Beispiel die aktuellen Verschlüsselungsverfahren. Trotz immer leistungsfähigerer Rechner ist es durch einfaches Vergrößern der Schlüssel immer wieder möglich, diese Algorithmen sicher zu halten. Zuletzt ist es natürlich wichtig zu wissen, ob ein entworfener Algorithmus auch in allen Fällen das tut, was er tun soll, d.h. ob er korrekt arbeitet. Der Nachweis der Korrektheit ist allgemein nicht möglich (vgl. Berechenbarkeit).

A. Fehler

In Algorithmen können drei verschiedene Arten von Fehlern auftreten, die unterschiedlich problematisch sind.

Zunächst muss der Prozessor die einzelnen Teile jeder Anweisung erkennen. Er benötigt also Kenntnisse über das Vokabular und die Grammatik der Sprache. Die Menge der grammatischen Regeln einer Sprache bezeichnet man als Syntax. Ein Algorithmus, der diese Regeln verletzt, besitzt einen syntaktischen Fehler. Solche Fehler ergeben sich häufig durch Tippfehler und können vom Prozessor vor der Ausführung der fehlerhaften Anweisung erkannt werden.

Jede Anweisung besitzt eine Bedeutung. So bedeutet etwa „Gesamtbetrag := Anzahl * Einzelbetrag“, daß die beiden Zahlen, die mit Anzahl und Einzelbetrag bezeichnet werden, zu multiplizieren sind und das Produkt die Zahl, die mit Gesamtkosten bezeichnet ist, ergibt. Die Bedeutung der Ausdrucksformen einer Sprache heißt Semantik. Es gibt Anweisungen, die syntaktisch korrekt, jedoch semantisch unsinnig sind. So macht eine Anweisung „Gib den 13. Monat des Jahres aus“ keinen Sinn, obwohl sie syntaktisch in Ordnung ist. Um syntaktische Fehler zu erkennen, muß der Prozessor die Bedeutung der Anweisung erkennen können. Weiß der Prozessor, daß ein Jahr nur 12 Monate besitzt, so kann er vor der Ausführung der Anweisung den Fehler entdecken. Andernfalls wird der Fehler erst bei der Ausführung selbst bemerkt.

Besonders schwer zu entdecken sind die logischen Fehler (und leider sind die meisten Fehler logische Fehler). Alle Anweisungen sind zwar syntaktisch und semantisch korrekt, aber der Algorithmus beschreibt den gewünschten Ablauf nicht richtig. So ist der Algorithmus „Berechne den Kreisumfang durch Multiplikation von π mit dem Radius“ syntaktisch und semantisch korrekt, aber das Ergebnis ist dennoch falsch. Ein Computer kann logische Fehler nicht entdecken, da er keine Vorstellung davon hat, was der Algorithmus tun soll. Sie zu erkennen ist schwierig, aber ein systematischer Entwurf hilft dabei, solche Fehler zu vermeiden.

B. Systematischer Entwurf

Der Entwurf von Algorithmen ist ein kreativer Prozess und weitaus schwieriger als die Programmierung eines Algorithmus in einer Programmiersprache. Leider gibt es keinen Algorithmus zum Entwurf von Algorithmen, aber es gibt einen Leitfaden, nachdem man vorgehen kann. Zunächst einmal sollten folgende Fragen beantwortet werden:

- Wie lautet das eigentliche Problem?

- Ist die Problemstellung klar und exakt?
- Hat das Problem bereits einen Namen?
- Was ist bekannt oder gegeben?
- Was ist unbekannt oder gesucht?
- Welche Bedingungen und Zusammenhänge bestehen zwischen den Ein- und Ausgabeobjekten?

Nur selten ist das Problem von vornherein klar. Daher müssen die Spezifikationen gemeinsam erarbeitet werden.

Wenn die grundsätzlichen Fragen geklärt sind, sollte geprüft werden,

- ob dasselbe oder ein ähnliches/vergleichbares Problem bekannt ist.
- ob ein allgemeineres Problem bekannt ist (wenn sich das Problem verallgemeinern läßt, ohne dass die Lösung erheblich schwerer wird, dann löse man das allgemeinere Problem).
- ob sich das Problem in ein oder mehrere einfachere Teilprobleme aufteilen lässt.

Um Fehler beim Entwurf möglichst von vornherein zu vermeiden, muss man alle denkbaren Umstände berücksichtigen. Dies ist bei komplexen Algorithmen sehr schwer. Deshalb ist es notwendig, den auszuführenden Prozess in kleinere Einzelschritte zu zerlegen, für die jeweils ein Algorithmus zur Lösung existiert, und die weniger umfangreich sind („divide et conquera“). Ist ein solcher Einzelschritt immer noch zu komplex, wird er einfach weiter zerlegt, solange, bis die einzelnen Schritte detailliert und präzise genug sind, um vom jeweiligen Prozessor ausgeführt werden zu können. Dieses Verfahren nennt sich schrittweise Verfeinerung und endet, wenn der Prozessor alle Schritte ausführen kann.

C. Die Güte von Algorithmen

Es gibt unterschiedliche Kriterien, anhand derer entschieden werden kann, was ein guter Algorithmus ist:

- Zeitbedarf
- Speicherbedarf
- Zeitbedarf für Entwicklung
- Übersichtlichkeit

Je nach Aufgabenstellung sind diese Kriterien unterschiedlich stark zu gewichten. Für den Algorithmus selbst sind sicher die Übersichtlichkeit und der Zeitbedarf für die Entwicklung am wichtigsten (dies bedeutet aber nicht, dass für das eigentliche Programm nicht auch eine Optimierung des Zeitbedarfs oder des Speicherbedarfs sinnvoll oder notwendig sein kann).