1 Einführung

In diesem Kapitel werden wir wichtige Themen der Informatik in einer ersten Übersicht darstellen. Zunächst beschäftigen wir uns mit dem Begriff *Informatik*, dann mit fundamentalen Grundbegriffen wie z.B. *Bits* und *Bytes*. Danach behandeln wir die Frage, wie Texte, logische Werte und Zahlen in Computern gespeichert werden. Wir erklären den Aufbau eines PCs und das Zusammenwirken von Hardware, Controllern, Treibern und Betriebssystem bis zur benutzerfreundlichen Anwendungssoftware. Viele der hier eingeführten Begriffe werden in den späteren Kapiteln noch eingehender behandelt. Daher dient dieses Kapitel als erster Überblick und als Grundsteinlegung für die folgenden.

1.1 Was ist „*Informatik*“?

Der Begriff *Informatik* leitet sich von dem Begriff *Information* her. Er entstand in den 60er Jahren. Informatik ist die Wissenschaft von der maschinellen Informationsverarbeitung. Die englische Bezeichnung für Informatik ist *Computer Science*, also die Wissenschaft, die sich mit Rechnern beschäftigt. Wenn auch die beiden Begriffe verschiedene Blickrichtungen andeuten, bezeichnen sie dennoch das Gleiche. Die Spannweite der Disziplin Informatik ist sehr breit, und demzufolge ist das Gebiet in mehrere Teilgebiete untergliedert.

1.1.1 Technische Informatik

Die *Technische Informatik* beschäftigt sich vorwiegend mit der Konstruktion von Rechnern, Speicherchips, schnellen Prozessoren oder Parallelprozessoren, aber auch mit dem Aufbau von Peripheriegeräten wie Festplatten, Druckern und Bildschirmen. Die Grenzen zwischen der Technischen Informatik und der Elektrotechnik sind fließend. An einigen Universitäten gibt es den Studiengang *Datentechnik*, der gerade diesen Grenzbereich zwischen Elektrotechnik und Informatik zum Gegenstand hat.

Man kann vereinfachend sagen, dass die Technische Informatik für die Bereitstellung der Gerätschaften, der so genannten *Hardware*, zuständig ist, welche die Grundlage jeder maschinellen Informationsverarbeitung darstellt. Naturgemäß muss die Technische Informatik aber auch die beabsichtigten Anwendungsgebiete der Hardware im Auge haben. Insbesondere muss sie die Anforderungen der Programme berücksichtigen, die durch diese Hardware ausgeführt werden sollen. Es ist ein Unterschied, ob ein Rechner extrem viele Daten in begrenzter Zeit verarbeiten soll, wie etwa bei der Wettervorhersage oder bei der Steuerung einer Raumfähre, oder ob er im kommerziellen oder im häuslichen Bereich einge-

setzt wird, wo es mehr auf die Unterstützung intuitiver Benutzerführung, die Präsentation von Grafiken, Text oder Sound ankommt.

1.1.2 Praktische Informatik

Die *Praktische Informatik* beschäftigt sich im weitesten Sinne mit den Programmen, die einen Rechner steuern. Im Gegensatz zur Hardware sind solche Programme leicht veränderbar, man spricht daher auch von *Software*. Es ist ein weiter Schritt von den recht primitiven Operationen, die die Hardware eines Rechners ausführen kann, bis zu den Anwendungsprogrammen, wie etwa Textverarbeitungssystemen, Spielen und Grafiksystemen, mit denen ein Anwender umgeht. Die Brücke zwischen der Hardware und der Anwendungssoftware zu schlagen, ist die Aufgabe der Praktischen Informatik.

Ein klassisches Gebiet der Praktischen Informatik ist der Compilerbau. Ein *Compiler* übersetzt Programme, die in einer technisch-intuitiven Notation, einer so genannten *Programmiersprache*, formuliert sind, in die stark von den technischen Besonderheiten der Maschine geprägte Notation der *Maschinensprache*. Es gibt viele populäre Programmiersprachen, darunter BASIC, Cobol, Fortran, Pascal, C, C++, C#, Java, Scala, JavaScript, PHP, Perl, LISP, ML und PROLOG. Programme, die in solchen *Hochsprachen* formuliert sind, können nach der Übersetzung durch einen Compiler auf den verschiedensten Maschinen ausgeführt werden oder, wie es im Informatik-Slang heißt, *laufen*. Ein Maschinenspracheprogramm läuft dagegen immer nur auf dem Maschinentyp, für den es geschrieben wurde.

1.1.3 Theoretische Informatik

Die *Theoretische Informatik* beschäftigt sich mit den abstrakten mathematischen und logischen Grundlagen aller Teilgebiete der Informatik. Theorie und Praxis sind in der Informatik enger verwoben, als in vielen anderen Disziplinen, theoretische Erkenntnisse sind schneller und direkter einsetzbar. Durch die theoretischen Arbeiten auf dem Gebiet der *formalen Sprachen* und der *Automatentheorie* zum Beispiel hat man das Gebiet des Compilerbaus heute sehr gut im Griff. In Anlehnung an die theoretischen Erkenntnisse sind praktische Werkzeuge entstanden. Diese sind selbst wieder Programme, mit denen ein großer Teil des Compilerbaus automatisiert werden kann. Bevor eine solche Theorie existierte, musste man mit einem Aufwand von ca. 25 *Bearbeiter-Jahren* (Anzahl der Bearbeiter * Arbeitszeit = 25) für die Konstruktion eines einfachen Compilers rechnen, heute erledigen Studenten eine vergleichbare Aufgabe im Rahmen eines Praktikums.

Neben den Beiträgen, die die Theoretische Informatik zur Entwicklung des Gebietes leistet, ist die Kenntnis der theoretischen Strukturen eine wichtige Schulung für jeden, der komplexe Systeme entwirft. Gut durchdachte, theoretisch abgesicherte Entwürfe erweisen sich auch für hochkomplexe Software als sicher und erweiterbar. Software-Systeme, die im Hauruck-Verfahren entstehen, stoßen immer bald an die Grenze, ab der sie nicht mehr weiterentwickelbar sind. Die Entwicklung von Software sollte sich an der Ökonomie der Theoriebildung in der Mathematik orientieren: möglichst wenige Annahmen, möglichst keine Ausnahmen. Ein wichtiger Grund etwa, warum die Konstruktion von Fortran-Compilern so kompliziert ist,

liegt darin, dass dieses Prinzip bei der Definition der Programmiersprache nicht angewendet wurde. Jeder Sonder- oder Ausnahmefall, jede zusätzliche Regel macht nicht nur dem Konstrukteur des Compilers das Leben schwer, sondern auch den vielen Fortran-Programmierern.

1.1.4 Angewandte Informatik

Die *Angewandte Informatik* beschäftigt sich mit dem Einsatz von Rechnern in den verschiedensten Bereichen unseres Lebens. Da in den letzten Jahren die Hardware eines Rechners für jeden erschwinglich geworden ist, gibt es auch keinen Bereich mehr, der der Computeranwendung verschlossen ist. Einerseits gilt es, spezialisierte Programme für bestimmte Aufgaben zu erstellen, andererseits müssen Programme und Konzepte entworfen werden, die in vielfältigen Umgebungen einsetzbar sein sollen. Beispiele für solche universell einsetzbaren Systeme sind etwa *Textverarbeitungssysteme* oder *Tabellenkalkulationssysteme* (engl. *spread sheet*). Angewandte Informatik nutzt heute jeder, der im *WWW* die aktuelle Tageszeitung liest, seine E-Mail erledigt, Bankgeschäfte tätigt, chattet, sich in sozialen Netzwerken tummelt, Musik herunterlädt, oder nur Filme anschaut.

Auch die Angewandte Informatik ist nicht isoliert von den anderen Gebieten denkbar. Es gilt schließlich, sowohl neue Möglichkeiten der Hardware als auch im Zusammenspiel von Theoretischer und Praktischer Informatik entstandene Werkzeuge einer sinnvollen Anwendung zuzuführen. Als Beispiel mögen die *SmartPhones, Organizer, Palmtops* und *Tablet-PCs* dienen, die im Wesentlichen aus einem Flüssigkristall-Bildschirm bestehen, den man mit einem Griffel oder einfach mit Fingergesten bedienen kann. Die Angewandte Informatik muss die Einsatzmöglichkeiten solcher Geräte, etwa in der mobilen Lagerhaltung, auf der Baustelle oder als vielseitiger, „intelligenter“ Terminkalender entwickeln. Die Hardware wurde von der Technischen Informatik konstruiert, die Softwaregrundlagen, etwa zur Handschrifterkennung, von der Praktischen Informatik aufgrund der Ergebnisse der Theoretischen Informatik gewonnen.

Wenn man im deutschsprachigen Raum auch diese Einteilung der Informatik vornimmt, so ist es klar, dass die einzelnen Gebiete nicht isoliert und ihre jeweiligen Grenzen nicht wohldefiniert sind. Die Technische Informatik überlappt sich stark mit der Praktischen Informatik, jene wieder mit der Theoretischen Informatik. Auch die Grenzen zwischen der Praktischen und der Angewandten Informatik sind fließend. Gleichgültig in welchem Bereich man später einmal arbeiten möchte, muss man auch die wichtigsten Methoden der Nachbargebiete kennen lernen, um die Möglichkeiten seines Gebietes entfalten und entwickeln zu lernen, aber auch um die Grenzen abschätzen zu können.

Neben der in diese vier Bereiche eingeteilten Informatik haben viele Anwendungsgebiete ihre eigenen Informatik-Ableger eingerichtet. So spricht man zum Beispiel von der *Medizinischen Informatik*, der *Wirtschaftsinformatik*, der *Medieninformatik*, der *Bio-Informatik*, der *Linguistischen Informatik*, der *Juristischen Informatik* oder der *Chemie-Informatik*. Für einige dieser Bereiche gibt es an Fachhochschulen und Universitäten bereits Studiengänge. Insbesondere geht es darum, fundierte Kenntnisse über das angestrebte Anwendungsgebiet mit grundlegenden Kenntnissen informatischer Methoden zu verbinden.

1.2 Information und Daten

Was tut eigentlich ein Computer? Diese Frage scheint leicht beantwortbar zu sein, indem wir einfach eine Fülle von Anwendungen aufzählen. Computer berechnen Wettervorhersagen, steuern Raumfähren, spielen Schach, machen Musik und erzeugen verblüffende Effekte in Kinofilmen. Sicher liegt hier aber nicht die Antwort auf die gestellte Frage, denn wir wollen natürlich wissen, *wie* Computer das machen. Um dies zu erklären, müssen wir uns zunächst einigen, in welcher Tiefe wir anfangen sollen. Bei der Erklärung des Schachprogramms wollen wir vielleicht wissen:

- Wie wird das Schachspiel des Computers bedient?
- Wie ist das Schachprogramm aufgebaut?
- Wie sind die Informationen über den Spielstand im Hauptspeicher des Rechners gespeichert und wie werden sie verändert?
- Wie sind die Nullen und Einsen in den einzelnen Speicherzellen organisiert und wie werden sie verändert?
- Welche elektrischen Signale beeinflussen die Transistoren und Widerstände, aus denen Speicherzellen und Prozessor aufgebaut sind?

Wir müssen uns auf *eine* solche mögliche Erklärungsebene festlegen. Da es hier um Informatik geht, also um die Verarbeitung von Informationen, beginnen wir auf der Ebene der Nullen und Einsen, denn dies ist die niedrigste Ebene der *Informationsverarbeitung*. Wir beschäftigen uns also zunächst damit, wie Informationen im Rechner durch Nullen und Einsen repräsentiert werden können. Die so repräsentierten Informationen nennen wir *Daten*. Die *Repräsentation* muss derart gewählt werden, dass man aus den Daten auch wieder die repräsentierte Information zurückgewinnen kann. Diesen Prozess der Interpretation von Daten als Information nennt man auch *Abstraktion*.

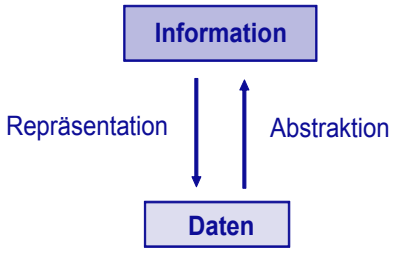

Abb. 1.1: Information und Daten

1.2.1 Bits

Ein *Bit* ist die kleinstmögliche Einheit der Information. Ein Bit ist die Informationsmenge in einer Antwort auf eine Frage, die zwei Möglichkeiten zulässt:

- ja oder nein,
- wahr oder falsch,
- schwarz oder weiß,
- hell oder dunkel,
- groß oder klein,
- stark oder schwach,
- links oder rechts.

Zu einer solchen Frage lässt sich immer eine *Codierung* der Antwort festlegen. Da es zwei mögliche Antworten gibt, reicht ein *Code* mit zwei Zeichen, ein so genannter *binärer Code*. Man benutzt dazu die Zeichen:

> `0` und `1`.

Eine solche Codierung ist deswegen nötig, weil die Information technisch dargestellt werden muss. Man bedient sich dabei etwa elektrischer Ladungen:

> `0` = ungeladen,
> `1` = geladen,

oder elektrischer Spannungen

> `0` = 0 Volt,
> `1` = 5 Volt,

oder Magnetisierungen

> `0` = unmagnetisiert,
> `1` = magnetisiert.

So kann man etwa die Antwort auf die Frage:

> *Welche Farbe hat der Springer auf F3?*

im Endeffekt dadurch repräsentieren bzw. auffinden, indem man prüft,

- ob ein Kondensator eine bestimmte Ladung besitzt,
- ob an einem Widerstand eine bestimmte Spannung anliegt oder
- ob eine bestimmte Stelle auf einer Magnetscheibe magnetisiert ist.

Da es uns im Moment aber nicht auf die genaue technische Realisierung ankommt, wollen wir die Übersetzung physikalischer Größen in Informationseinheiten bereits voraussetzen und nur von den beiden möglichen Elementarinformationen 0 und 1 ausgehen.

1.2.2 Bitfolgen

Lässt eine Frage mehrere Antworten zu, so enthält die Beantwortung der Frage mehr als ein Bit an Information. Die Frage etwa, aus welcher Himmelsrichtung, Nord, Süd, Ost oder West, der Wind weht, lässt 4 mögliche Antworten zu. Der Informationsgehalt in der Beantwortung der Frage ist aber nur 2 Bit, denn man kann die ursprüngliche Frage in zwei andere Fragen verwandeln, die jeweils nur zwei Antworten zulassen:

1. Weht der Wind aus einer der Richtungen Nord oder Ost (ja/nein)?
2. Weht der Wind aus einer der Richtungen Ost oder West (ja/nein)?

Eine mögliche Antwort, etwa *ja* auf die erste Frage und *nein* auf die zweite Frage, lässt sich durch die beiden Bits

1 0

repräsentieren. Die Bitfolge `10` besagt also diesmal, dass der Wind aus Norden weht. Ähnlich repräsentieren die Bitfolgen

0 0 ↔ Süd **0 1** ↔ West **1 0** ↔ Nord **1 1** ↔ Ost.

Offensichtlich gibt es genau 4 mögliche Folgen von 2 Bit. Mit 2 Bit können wir also Fragen beantworten, die 4 mögliche Antworten zulassen.

Lassen wir auf dieselbe Frage (Woher weht der Wind?) auch noch die Zwischenrichtungen Südost, Nordwest, Nordost und Südwest zu, so gibt es 4 weitere mögliche Antworten, also insgesamt 8. Mit einem zusätzlichen Bit, also mit insgesamt 3 Bits, können wir alle 8 möglichen Antworten darstellen. Die möglichen Folgen aus 3 Bits sind:

000, 001, 010, 011, 100, 101, 110, 111.

und die möglichen Antworten auf die Frage nach der Windrichtung sind:

Süd, West, Nord, Ost, Südost, Nordwest, Nordost, Südwest.

Jede beliebige eindeutige Zuordnung der Himmelsrichtungen zu diesen Bitfolgen können wir als Codierung von Windrichtungen hernehmen, zum Beispiel:

000 = Süd	**100** = Südost
001 = West	**101** = Nordwest
010 = Nord	**110** = Nordost
011 = Ost	**111** = Südwest

Offensichtlich verdoppelt jedes zusätzliche Bit die Anzahl der möglichen Bitfolgen, so dass gilt:

Es gibt genau 2^n verschiedene Bitfolgen der Länge n.