Informatik I

Dozent: Prof. Dr. Martin Wirsing

Betreuer: Dr. Bernhard Reus,
Dr. Piotr Kosiuczenko,
Dirk Pattinson

Zeit: Di, Fr 11-13, HS 122

Kapitel 1

Einleitung

1.1 Was ist Informatik?

Informatik ist ein Kunstwort, das zu Beginn der 60er Jahre zur Bezeichnung einer sich neu entwickelnden Disziplin geschaffen wurde. Es setzt sich aus Bestandteilen der beiden Worte Information und Mathematik zusammen. Darin kommt zum Ausdruck, daß Informatik die Wissenschaft von der Informationsverarbeitung ist und eine große Nähe zur Mathematik hat. Anders als im englischen Sprachraum, wo man für diese Disziplin das Wort *Computer Science* verwendet, kommt im Begriff Informatik das Wort Computer nicht vor. Dennoch spielen Computer in der Informatik eine wichtige Rolle. Genauer kann die Disziplin durch Begriffe wie *Information, Informationssystem, Rechenanlage, Computer, EDVDaten, Datenstruktur, Programm, Programmiersprache* usw. inhaltlich umrissen werden.

Es gibt verschiedene Versuche zur *Definition dieser Disziplin*. Eine erste Möglichkeit ist, die Informatik wie folgt zu definieren: Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Computern (vgl. DUDEN Informatik).
Die *Gesellschaft für Informatik*, www.gi-ev.de (Studien- und Forschungsführer Informatik, Springer Verlag) definiert: Informatik ist die Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung und Übermittlung von Informationen. Die amerikanische Computergesellschaft Association of Computing Maschinery (ACM, www.acm.org) schreibt in ihrem Mitteilungsblatt Communications of the ACM 1986:
"Computer Science is the systematic study of algorithms and data structures specifically

1. their formal properties,

2. their mechanical and linguistic realizations, and

3. their applications."

Dabei wird betont, daß diese Reihenfolge eine Gewichtung enthält, die wesentlich ist. Reiht man die drei Charakteristika des Fachs *Computer Science* in der Anordnung zunächst 2, dann 1, dann 3 - steht also die Realisierung von Algorithmen im Vordergrund -, so wird von *Computer Engineering* gesprochen. Reiht man die drei Charakteristika in der Reihenfolge 3, 2, 1 - das heißt, stehen die Anwendungen im Vordergrund -, so spricht man von *Information Systems*.
Als wichtigste Bestandteile des Kerns eines jeden Computer-Science-Studiums werden genannt:

- Principles of Computer Organization
- Algorithms
- Theory of Computation
- Principles of Programming Languages.

Diese Vorstellung deckt sich weitgehend mit dem, was an deutschen Universitäten im Grundstudium der Informatik üblich ist. Deutlich wird in der Beschreibung der amerikanischen Computergesellschaft auch, daß, obwohl im Namen Computer Science der Begriff Computer vorkommt, der zentrale Begriff der Informatik der des Algorithmus ist.

Es ist meistens üblich, die Informatik als Wissenschaft in verschiedene **Teilgebiete** zu untergliedern, und zwar in die folgenden:

- Theoretische Informatik: Hier werden mathematische Modelle von Computern entwickelt, einschließlich von Hilfsmitteln zu ihrer präzisen Beschreibung. Wichtige Begriffe sind: Berechenbarkeit, Entscheidbarkeit, Komplexität.

- Technische Informatik: Sie beschäftigt sich mit der inneren Struktur und dem Bau von Computern und deren Komponenten sowie mit allen damit zusammenhängenden technischen Fragen.

- Praktische Informatik: Sie umfaßt die Prinzipien und die Techniken der Programmierung, also insbesondere das, was man auch als Software Engineering bezeichnet. Weitere Beispiele für Themen, die in der praktischen Informatik behandelt werden, sind: Datenbanken, Betriebssysteme, Compilerbau, Bildverarbeitende Systeme, Spracherkennung, Expertensysteme u.a.

- Angewandte Informatik: Sie bildet die Brücke von den Computerwissenschaften, also der Informatik im engeren Sinne, zu den Problemen der realen Welt. Hier gliedern sich alle sogenannten Bindestrich-Informatiken ein, wie zum Beispiel Wirtschafts-Informatik, Rechts-Informatik, Medizinische Informatik usw.

Neuerdings beschäftigt man sich zunehmend auch mit den Auswirkungen der Informationstechnik auf die Gesellschaft und faßt die Untersuchungen hierzu unter dem Begriff Informatik und Gesellschaft zusammen.

Das, was man unter Informatik versteht, kann man in solchen Definitionen jedoch niemals ganz und endgültig fassen. Schließlich ändert sich die Beschreibung der Definitionen einer Wissenschaft, wie in anderen Fällen auch, über die Zeit, so daß man besser eine operationale Definition nehmen sollte, um als Informatik das zu bezeichnen, was die Gemeinschaft der Informatiker an den Hochschulen und im Beruf an Tätigkeiten ausübt.

1.2 Historische Entwicklung

Obwohl die Informatik als wissenschaftliche Disziplin noch nicht einmal 30 Jahre alt ist, ist die Beschäftigung mit ihren zentralen Inhalten, also mit Algorithmen aller Art und auch der Bau von Computern, wesentlich älter. Wir skizzieren zunächst die historische Entwicklung kurz (vgl. DUDEN Informatik).

- Altertum-Mittelalter: Verwendung des Abakus (lat.: abax, Tafel, chin.: suan-pan, Brett mit verschiebbaren Kugeln) als Hilfsmittel für die vier Grundrechenarten.

- 9. Jh.: Der arabische Mathematiker und Astronom Ibn Musa Al-Chwarismi schreibt das

Lehrbuch "Kitab al jabr w'almuqabala" ("Regeln der Wiedereinsetzung und Reduktion"). Das Wort 'algorithmus" geht auf seinen Namen zurück.

- 1202: Der italienische Mathematiker und Kaufmann Leonardo Fibonacci schreibt das „Liber Abaci" (Buch des Rechnens) und führt das Dezimalsystem im Westen ein.

- 1547: Adam Riese (1492-1559) veröffentlicht ein Rechenbuch, in dem er die Rechengesetze des aus Indien stammenden Dezimalsystems (5. Jh. n. Chr.) beschreibt. Im 17. Jahrhundert setzt sich das Dezimalsystem in Europa durch. Nun ist eine Automatisierung des Rechenvorgangs möglich.

- 1623 : Wilhelm Schickard (1592-1635) konstruiert für seinen Freund Kepler (157l-1630) eine Maschine, die addieren, subtrahieren, multiplizieren und dividieren kann. Sie bleibt unbeachtet.

- 1641: Blaise Pascal (1623-1662) konstruiert eine Maschine, mit der man sechsstellige Zahlen addieren kann.

- 1674: Gottfried Wilhelm Leibniz (1646-1716, Philosoph, Rechtsgelehrter, Politiker, Geschichts- und Sprachforscher, Naturwissenschaftler und Mathematiker) konstruiert eine Rechenmaschine mit Staffelwalzen für die vier Grundrechenarten. In diesem Zusammenhang befaßt er sich auch mit der binären Darstellung von Zahlen.

- 1774: Philipp Matthäus Hahn (1739-1790) entwickelte eine mechanische Rechenmaschine, die erstmals zuverlässig arbeitet.

- Ab 1818: Rechenmaschinen nach dem Vorbild der Leibnizschen Maschine werden serienmäßig hergestellt und dabei ständig weiterentwickelt.

- 1838: Charles Babbage (1792-1871) plant eine Maschine, die „analytical Engine“, bei der die Reihenfolge der einzelnen Rechenoperationen durch nacheinander eingegebene Lochkarten gesteuert wird. Die Programmiersprache ADA wurde nach dem Vornamen der Assistentin von Charles Babbage, Gräfin Ada Augusta von Lovelace, benannt.

- 1886: Hermann Hollerith (1860-l929) entwickelt in den USA elektrisch arbeitende Zählmaschinen für Lochkarten, mit denen die statistischen Auswertungen der Volkszählungen vorgenommen werden.

- 1934: Konrad Zuse (1910-1995) beginnt mit der Planung einer programmgesteuerten Rechenmaschine. Sie verwendet das binäre Zahlensystem und die halblogarithmische Zahlendarstellung.

- 1937: Die mechanische Anlage Z 1 von Zuse ist fertig.

- 1941: Die elektromechanische Anlage Z 3 von Zuse ist fertig. Dies ist der erste funktionsfähige programmgesteuerte Rechenautomat. Das Programm wurde mit Lochstreifen eingegeben. Die Anlage verfügt über 2000 Relais und eine Speicherkapaziät von 64 Worten a 22 Bit. Multiplikationszeit: etwa 3 s.

- 1944: Howard H. Aiken (1900-1973) erstellt in Zusammenarbeit mit der Harvard-University und der Firma IBM die teilweise programmgesteuerte Rechenanlage MARK I. Additionszeit 1/3 s, Multiplikationszeit: 6 s.

- 1946: J. P. Eckert und J. W. Mauchly stellen die ENIAC (Electronic Numerical Integrator and Automatic Calculator) fertig. Dies ist der erste voll elektronische Rechner (18.000 Elektronenröhren). Multiplikationszeit: 3 ms.

- 1946-1952: Auf der Grundlage der Ideen John v. Neumanns (1903-1957) (Einzelprozessor, Programm und Daten im gleichen Speicher; Von-Neumann-Rechner) und seiner Kollegen am Institute of Advanced Study at Princeton (H.H.Goldstine, A.W.Burks) werden weitere Computer in Universitätslabors entwickelt ("Pionierzeit").

- 1949: M.V.Wills (University of Manchester) stellt mit der EDSAC (Electronic Delay Storage Automatic Calculator) den ersten universellen Digitalrechner (gespeichertes Programm) fertig.

- Ab 1950: Industrielle Rechnerentwicklung und -produktion.

- 1949-1955: Bau der PERM durch F. L. Bauer, K. Samelson unter Leitung von R. Sauer

Die **industrielle Rechnerentwicklung** wird üblicherweise in verschiedene Generationen eingeteilt. Sie können wie folgt charakterisiert werden:

1. Generation (bis Ende der 50er Jahre): hier dienen Elektronenröhren als Schaltelemente (Geschwindigkeit: etwa 1000 Additionen/s);

2. Generation (bis Ende der 60er Jahre): Halbleiterschaltkreise werden als Schaltelemente verwendet (Transistoren, Dioden; Geschwindigkeit: etwa 10.000 Additionen/s);

3. Generation (seit Mitte der 60er Jahre): teilintegrierte Schaltkreise (Mikromodultechnik mit ca. 20 Schaltelementen/qcm; Geschwindigkeit: etwa 500.000 Additionen/s);

4. Generation (seit Anfang der 70er Jahre): hochintegrierte Schaltkreise bis zu 1000 Schaltelementen/qcm; Geschwindigkeit: etwa 10 Mio. Additionen/s;

5. Generation (seit Anfang der 80er Jahre): höchstintegrierte Schaltkreise; mehrere Millionen Schaltelemente/qcm; mehrere Prozessoren auf einem Chip; Geschwindigkeitssteigerung durch Parallelisierung: bis zu 100 Mio. Additionen/s und mehr.

6. Generation (Zukunft?): Quanten-Computer, rechnet mit Überlagerungen von Quantenzuständen, Explosion der Berechnungsmöglichkeiten.

1.3 Algorithmenbegriff und Programmiersprachen

1.3.1 Empirisch/intuitiver Algorithmenbegriff

Im täglichen Leben begegnen uns Algorithmen als Handlungsanweisungen aller Art, wie zum Beispiel die folgenden:

- Ärztliche Verordnung: Nimm dreimal täglich 15 Tropfen Asperix vor den Mahlzeiten.

- Waschanleitung: Bei 60 Grad waschen; Waschmittelzugabe in Abhängigkeit von der Wasserhärte nach Angaben des Herstellers.

- Einfahrvorschrift für Autos: Im 2. Gang nicht über 50 km/h, im 3. Gang nicht über 80 km/h, im 4. Gang nicht über 120 km/h; nach 1000 gefahrenen km Motor- und Getriebeöl wechsel.

- Spielregel: ... bei einer 6 darf noch einmal gewürfelt werden ...

- Koch- oder Backrezepte

usw.

Beispiel aus der Küche

Pfannkuchen (Abstrakt)

Mehl, Milch, 2 Eier, eine Prise Salz und Zucker
Mischen, in Pfanne geben, auf mittlerer Hitze stocken lassen.

Pfannkuchen (Detailliert)

Mischen Sie 100 g Mehl
100 ml Milch
2 Eier
1 g Salz
10 g Zucker

mit dem Handmixer verrühren, lassen Sie Teig 10 Min. quellen. Zerlassen Sie Butter in einer Pfanne, geben Sie eine Portion Teig hinein, lassen Sie Teig auf Stufe 5 für 2 Min. stocken, Pfannkuchen wenden, nochmals 2 Min. in der Pfanne lassen, auf einen Teller geben, servieren.

Betrachtet man diese Alltagsbeispiele, so kann man feststellen, daß offensichtlich zwischen dem Text einer Handlungsanweisung und dem, der die Anweisung ausführt, also dem Bearbeiter (in der Computersprache spricht man vom Prozessor), unterschieden wird, und daß schließlich auch der Vorgang selbst, d. h. die Ausführung der Handlungsanweisung, von dem Text der Handlungsanweisung zu unterscheiden ist (das entsprechende Analogon in der Computersprache nennt man einen *Prozeß*).
Einzelne Anweisungen werden stets in bestimmter Reihenfolge ausgeführt. Diese kann abhängen von der Reihenfolge, in der die Anweisungen aufgeschrieben sind: Die Reihenfolge kann also mit der textuellen Reihenfolge der Beschreibung der Handlungsanweisungen übereinstimmen. Sie kann aber auch von Bedingungen abhängig gemacht werden, von Zwischenresultaten usw., wie das Beispiel, ". .. wenn eine 6 gewürfelt wird, darf noch einmal gewürfelt werden .. . ", zeigt. Bisweilen ist es auch erlaubt, Handlungsanweisungen, wie man sagt, nebenläufig, d. h. nicht sequentiell-nacheinander, sondern parallel-gleichzeitig, auszuführen,

d. h. die zeitliche Reihenfolge wird dann nicht festgelegt. Schließlich wird bei allen, auch bei Alltagsanweisungen, ein Unterschied zwischen den Daten und ihrer Beschreibung im Text gemacht. Also denken Sie etwa an das Beispiel des Backrezeptes: Hier ist die Beschreibung der Zutaten in einem Kochrezept natürlich von den Zutaten selbst, die zur Herstellung eines bestimmten Produkts benutzt werden, zu unterscheiden. Alle diese Dinge finden sich auch bei Algorithmen wieder.

Bevor wir eine vorläufige und intuitive Definition des Algorithmenbegriffs geben können, wollen wir die wichtigsten Merkmale von Algorithmen an einigen Beispielen herauspräparieren. Betrachten Sie etwa die Anweisung in der folgenden Abbildung. Sie soll eine Vorschrift bezeichnen, die keine endliche Länge hat. Eine solche Vorschrift werden wir nicht als einen Algorithmus bezeichnen.
Ein Beispiel für ein Verfahren, das zwar endlich beschrieben ist, aber vielleicht niemals abbricht, ist das folgende:

Wiederholen Sie die Schritte 1, 2 und 3 solange, bis Sie einen Gewinn von einer Million Mark erzielt haben:

1. Lottoschein ausfüllen.
2. Schein bei einer Annahmestelle abgeben.
3. Auf die nächste Ziehung warten und Gewinn kassieren.

Back- und Kochrezepte enthalten häufig Anweisungen, bei denen die Reihenfolge der Ausführung nicht eindeutig festgelegt ist, wie etwa in folgendem Beispiel:

Singen Sie:

Ein Mops kam in die Küche und stahl dem Koch ein Ei; da nahm der Koch den Löffel und schlug den Mops entzwei. Es kamen viele Möpse und gruben ihm ein Grab und setzten einen Grabstein, auf dem geschrieben stand:

Ein Mops kam in die Küche und stahl dem Koch ein Ei; da nahm der Koch den Löffel...

Abbildung: Bsp. für eine Vorschrift, die keine endliche Länge hat.

"Die Suppe aus der Dose nach Vorschrift zubereiten. Sie können zum Schluß noch einige Spargelstückchen hinzugeben. Den Schinken in Streifen schneiden..."

Hier wird nicht gesagt, in welcher Reihenfolge man die hier angegebenen Anweisungen durchführen muß, sondern nur in loser Form gesagt, was am Schluß gemacht werden soll. Manchmal ist die Wirkung einer Anweisung nicht eindeutig, wie in dem nachfolgend zitierten

Beispiel:

Ein Regierungsmitglied steht auf dem Bahnsteig und verabschiedet einen Staatsgast. Nachdem dieser eingestiegen ist, ruft der Bahnhofsvorsteher:

". . . bitte zurücktreten! "

Diese Aufforderung kann auch eine völlig andere Bedeutung als die durch den geschilderten Kontext nahegelegte haben. Schließlich wird manchmal durchaus gesagt, was die Wirkung einer Anweisung sein soll, aber nicht, wie sie erzielt werden kann. Denken Sie etwa an das Beispiel der Umwandlung gegebener Temperaturwerte in Grad Celsius in solche in Grad Fahrenheit. Gegeben ist ein Temperaturwert x: in Grad Celsius als reelle Zahl auf zwei Dezimalstellen genau. Gesucht ist der x entsprechende Temperaturwert y, ebenfalls als reelle Zahl auf zwei Dezimalstellen genau. Diese Vorschrift ist ganz eindeutig, dennoch sagt sie nicht, wie der gesuchte Temperaturwert ermittelt werden kann.
(Wir wissen natürlich, daß $y = x - 9/5 + 32$ ist; das wäre eine solche Vorschrift, die zeigt, wie der gesuchte Wert y berechnet werden kann).

Fassen wir diese Diskussion zusammen und leiten daraus eine **intuitive Definition des Algorithmenbegriffs** ab:
Ein *Algorithmus* ist ein Verfahren mit einer präzisen (d.h. in einer genau festgelegten Sprache abgehaltenen), endlichen Beschreibung (statische Finitheit) unter Verwendung effektiver (d.h. tatsächlich ausführbarer) elementarer Verarbeitungsschritte. Zu jedem Zeitpunkt der Abarbeitung des Algorithmus hat der Algorithmus nur endlich viele Ressourcen belegt (dynamische Finitheit).
Ein Algorithmus liefert eine Funktion (Abbildung), die zu jeder zulässigen Eingabe die durch den Algorithmus definierte Ausgabe festlegt.

Ein Algorithmus heißt für eine Eingabe
- *terminierend*, wenn er für alle zulässigen Schrittfolgen stets nach endlich vielen Schritten endet,
- *deterministisch*, wenn in der Auswahl der Verarbeitungsschritte keine Freiheit besteht,
- *determiniert*, wenn das Resultat eindeutig bestimmt ist,
- *sequentiell*, wenn die Schritte stets hintereinander ausgeführt werden,
- *parallel (nebenläufig)*, wenn gewisse Verarbeitungsschritte nebeneinander ausgeführt werden.

Bemerkung:
(1) Diese Definition ist nicht exakt und hängt vom Verständnis der verwendeten Begriffe ab. Was heißt zum Beispiel effektiv ?
(2) Auch beim Algorithmus unterscheidet man zwischen dem Begriff – ein Verfahren – und seiner Beschreibung.
(3) Algorithmen lösen eine Klasse von Aufgaben – die im allgemeinen durch Parameter bestimmt ist. Auf Eingaben liefern Algorithmen Resultate. Diese sind i.a. Repräsentationen von Informationen oder Folgen von Anweisungen.
(4) Wichtig ist der Unterschied zwischen
- der Aufgabenstellung und
- der Art und Weise, wie Aufgaben gelöst werden.

Dabei unterscheidet man zwischen den zur Verfügung stehenden elementaren Arbeits-

schritten (i.a. geg. durch die verwendete Programmiersprache und der Beschreibung der Auswahl der einzelnen Schritte, dem Programm).

(5) Es gibt zu jeder Aufgabe eine oder viele mögliche algorithmische Lösungen. Zu jedem Programm gibt es unendlich viele äquivalente Programme – in jeder üblichen Programmiersprache, die einen Arbeitsschritt für “Nichtstun” bereitstellt.

(6) In der Informatik spielen viele nicht-terminierende Algorithmen eine große Rolle. Sie werden beispielsweise zur Prozeßsteuerung, Datenübertragung in Netzen und Mensch-Maschine Kommunikation benutzt. (Man spricht in diesem Kontext auch von reaktiven Systemen.)

Eine intuitive Vorstellung des Algorithmenbegriffs reicht für die meisten Überlegungen, die wir in dieser Vorlesung anstellen, völlig aus, und zwar deshalb, weil wir meist konstruktiv vorgehen, d. h. wir geben ein Reihe von Algorithmen an und müssen uns dann nur davon überzeugen, daß die Algorithmen und ihre Darstellung die hier genannten Eigenschaften erfüllen. Die Situation wird erst dann völlig anders, wenn man von bestimmten Problemen behauptet, daß sie für eine algorithmische Lösung - und damit letztlich für die Lösung mit Hilfe von Computern - prinzipiell unzugänglich sind. Eine solche Aussage bezieht sich ja auf die Menge aller möglichen und denkbaren Algorithmen. Darüber läßt sich präzis nur dann argumentieren, wenn man eine im mathematischen Sinne exakte Definition des Algorithmenbegriffs zugrundelegt.

Definition
(1) Die Beschreibung eines Algorithmus in einer formal geschriebenen Sprache heißt *Programm*.
(2) Die formale Sprache heißt *Programmiersprache*.
(3) Eine Funktion heißt *berechenbar*, wenn sie sich durch einen Algorithmus realisieren läßt.

Bemerkung: Nicht alle Funktionen sind berechenbar.

1.3.2 Programmiersprachen

Parallel zu dieser technischen Entwicklung der Rechner vollzog sich auch die Entwicklung der Programmiersprachen, die charakterisiert ist durch eine zunehmende Anhebung der Sprachebene, d. h. weg von hardwarenahen hin zu immer mehr problemorientierten Sprachen. Hier kann man etwa folgende Generationen unterscheiden:

1. Generation: Programmierung im Maschinencode;

2. Generation: Assemblersprachen und erste Entwicklung höherer Programmiersprachen wie
FORTRAN, ALGOL und COBOL;

3. Generation: Entwicklung von Betriebssytemen mit Dialogbetrieb, Datenbanken, Methoden der strukturierten Programmierung, Programmiersprache Pascal;

4. Generation: verteilte Systeme, Rechnernetze, Kommunikationsfähigkeit, gute Arbeits- und Programmierumgebungen ;

5. Generation: Wissensverarbeitung, automatisches Schlußfolgern, deduktive Datenbanken, Expertensysteme, PROLOG.

Heute gibt es eine unübersehbare Fülle von Programmiersprachen, sicher mehrere tausend, für die verschiedensten Anwendungsgebiete. Nur wenige Programmiersprachell haben jedoch als universell verwendbare Sprachen einen hohen Verbreitungsgrad in der industriellen Praxis erreicht.

Es ist üblich, die Sprachen nach Programmierstilen zu unterteilen.

Beispiel

- Java-Programm:

```
public static int ggt (int x, int y) {
        while (x != y) {
                if (x > y) x = x - y;
                else       y = y - x;
        }
        return x;
}
```

- SML-Programm:

```
fun ggt x y =
        if x = y then x
        else if x < y then ggt y x
        else ggt (x-y) y;
```

Table 1:

Theorie	Programmierstiel	Sprachen
Funktionen	Funktionale Programmierung	ML, Miranda, Haskell
Prädikatenlogik	Logische Programmierung	Prolog und Varianten
Mengen	Mengen-orientierte Programmierung	SETL
Anweisungen	Imperative Programmierung	Basic, Algol, FORTRAN; PASCAL; Modula, C
„Objekte“	Objekt-orientierte Programmierung	Eiffel, Oberon, C++, Java
Datenbanken	„Query“-Programmierung	SQL, QUEL
Datenfluß	Datenflußprogrammierung	Lucid, Val, Id

Die Zuordnung einer Sprache zu einer dieser Klassen ist nicht immer eindeutig möglich. So haben beispielsweise die objektorientierten Sprachen C++ und Java einen imperativen Sprachkern und könnten so ebenso der Klasse der imperativen Sprachen zugeordnet werden. Die vier

Sprachklassen kennzeichnen daher eher einen gewissen Programmier-Stil und weniger die Ausdrucksfähigkeit einer bestimmten Sprache.