Einführung in die Informatik I (autip)

Dr. Stefan Lewandowski

Fakultät 5: Informatik, Elektrotechnik und Informationstechnik
Abteilung Formale Konzepte
Universität Stuttgart

17. Oktober 2007

- `www.fmi.uni-stuttgart.de/fk/lehre/ws07-08/autip1/`
- Vorlesung: Mittwochs 14:00–15:30, Raum V38.03
- Übung: Mittwochs 11:30–13:00, Raum 0.108
  - Wöchentliche Übungsblätter ab 24.10.
  - Abgabe und Bearbeitung über das System eClaus `http://eclaus.informatik.uni-stuttgart.de`
  - Eintragen in die Übungsgruppe:
    username: `online-Info1autip` ; password: `lewand07`
  - eClaus-Account wird eingerichtet, Passwort per E-Mail zugesendet
  - ab dann einloggen mit Ihrer Matrikelnummer und dem zugesandten Passwort
- Scheinbedingungen:
  - 50% der erreichbaren Punkte
    14 Übungsblätter a 20 Punkte
  - Bestehen von 2 der 3 Testklausuren
    voraussichtliche Termine: 21.11., 19.12., 6.2.
  - aktive, regelmäßige Teilnahme in den Übungen

Informatik im Studiengang Autip

- 1. Semester: Einführung in die Informatik 1 : 2V + 2Ü
- 2. Semester: Einführung in die Informatik 2 : 4V + 2Ü
- ca. 15.8.2008: Klausur Einführung in die Informatik 1+2
- Zulassungsvoraussetzung: Übungsschein in Informatik 1 oder 2

- 3. Semester: Einführung in die Informatik 3 : 3V + 2Ü
- ca. 15.3.2009: Klausur Einführung in die Informatik 3

- weitere Vorlesungen im Hauptstudium

Definition

Informatik ist alles, was mit Computern zu tun hat

falsch! – Definition aus dem Duden Informatik:

Informatik ist die Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen, besonders der automatischen Verarbeitung mithilfe von Digitalrechnern (Computer).

Lernziele sind also insbesondere **nicht** der Beherrschung von Windows oder Linux und den darauf laufenden Programmen.

Lernziele

- Grundkonzepte der Informatik:
  - Modellbildung und Abstraktion
  - Herangehensweisen zur Problemlösung
  - Algorithmen: Entwicklung und Bewertung
  - Standardalgorithmen und -datenstrukturen
- Kenntnis einer höheren Programmiersprache (aktiv, nicht nur passiv!), hier : Ada 95
- Am Rande gestreift:
  - ungefähre Vorstellung der Computerhardware
  - Aspekte des Software-Engineerings
- auch(!): soziale Kompetenzen, Softskills:
  - Teamarbeit
  - realistische Einschätzung der eigenen Leistung
  - ...

Materialien

Selbstständiges Arbeiten ist notwendig

- Erlernen von Ada, Programmierpraxis gewinnen
- Nachlesen der in der Vorlesung angesprochenen theoretischen Konzepte
- Unterlagen zur Hauptfach-Vorlesung Informatik (Prof. Levi)
  `http://www.ipvs.uni-stuttgart.de/abteilungen/bv/lehre/lehrveranstaltungen/vorlesungen/WS0708/Informatik_I/de`

Literatur

Die Literatur zu Ada 95 ist fast ausschließlich Englisch und richtet sich in der Regel an Menschen mit einigen Programmierkenntnissen in anderen Hochsprachen. Zwei Standardwerke seien hier genannt:

> J.G.P. Barnes, Programming in Ada95, 2. Auflage, Addison-Wesley, 1998
>
> M. Nagl, Softwaretechnik mit Ada 95, 2. Auflage, vieweg, 2003

Eine Sammlung von Links zu Ada 95 findet man auf der Ada-Seite der Fachschaft Informatik und Softwaretechnik

> `http://www.ada-lernen.de.vu` $\rightsquigarrow$ Ada

Ein empfehlenswertes englisches Buch für Anfänger

> J. Skansholm, Ada 95 from the beginning, 3. Auflage, 1998

Der Rechner und wir

- Zum Bearbeiten der Aufgaben Zugang zu Rechner unbedingt notwendig!
- Zugang über RUS oder Account im Poolraum der Informatik (Universitätsstr. 38) beantragen.
- Bequemer: eigener Rechner
  - Wesentliche Voraussetzung: Ada-Übersetzter, ist für Windows/Linux (kostenlos) verfügbar ($\rightsquigarrow$ Ada-CD-Rom, siehe Link auf der Vorlesungsseite).
  - Finden Sie sich so in Arbeitsgruppen zusammen, dass pro Gruppe ein Rechner (Windows/Linux) vorhanden ist.

Beispielproblem

Als Beispiel für die Aufgabenfelder der Informatik betrachten wir ein System zur Wettervorhersage. Eine Herausforderung für die „systematische und automatisierte Informationsverarbeitung“: wir haben

> Eingabedaten: eine Flut von Messwerten, die uns die aktuelle Wettersituation und deren bisherige Entwicklung liefern

und sollen daraus

> Ausgabedaten: die Wettervorhersage für morgen

gewinnen.

... Worauf warten wir? Ab an den Rechner! ...

STOPP! So nicht! (gilt auch für kleine Programme)

Modellbildung

Um unsere Aufgabe mit dem Rechner angehen zu können, müssen wir ein Modell des relevanten Teils der Realität entwickeln – Stichworte sind hier Abstraktion und Formalisierung.

- Für viele Problemstellungen liefert die Mathematik einen mächtigen Satz Werkzeuge zur präzisen Beschreibung der vorkommenden Größen und deren Beziehungen. Im Fall der Wettervorhersage lassen sich etwa Strömungsvorgänge mittels (recht ekliger) Differentialgleichungen beschreiben.
- Außer Mathematik brauchen wir bei diesem Schritt i.d. Regel Hilfe aus der Anwendungsdisziplin, hier aus der Meteorologie, aus der Physik und möglicherweise aus weiteren Disziplinen.
- Am Ende der Modellbildung ist aus dem unpräzisen Szenario („das Wetter“) eine präzise Beschreibung (in Größen wie „Temperaturverteilung im Gebiet XY“) geworden, die als Grundlage für die weiteren Arbeiten fassbar ist.

Modellbildung

- Meist wird in diesem Prozess erst klar, was der Kunde eigentlich will: vielleicht wollte er gar kein Programm, das die Temperaturverteilung vorherberechnet, sondern nur eines, das Bauernregeln auswertet?
- Formalisierung und Abstraktion sind wichtige Säulen des Problemlösungsprozesses (auch wenn der Sinn bei den kleinen Beispielen, wie wir sie in den Übungen behandeln können, vielleicht nicht so offensichtlich ist)!

Computergerechte Repräsentation

Bei der Formalisierung hatten wir zwar schon im Hinterkopf, dass mal ein Programm entstehen soll, aber im Mittelpunkt der Überlegungen stand die Beschreibung des Problems, nicht seine Lösung: eine Differentialgleichung ist kein Programm!
Der nächste Schritt bewegt sich in Richtung Rechner das formalisierte Problem wird in eine für den Rechner geeignete Form aufbereitet:

- Was für Daten sollen gespeichert werden $\rightsquigarrow$ Datenstrukturen
- Was soll mit diesen Daten passieren $\rightsquigarrow$ Algorithmen (Berechnungsvorschriften)

Auswahl/Entwicklung von Datenstrukturen

Wie sieht eine geeignete Repräsentation der Problemgrößen auf einem Rechner aus?

- Unser Rechner hat z.B. einen zwar großen, aber endlichen Speicher: schon bei der Darstellung (reeller) Zahlen sind Näherung notwendig.
- Wesentlich problematischer: Feldgrößen (etwa der Temperaturverteilung über einem Gebiet); prinzipiell unendlich viele Funktionswerte. Hier müssen noch mal Mathematiker und Informatiker (und der Kunde) gemeinsam ran, um eine geeignete diskretisierte (endliche) Näherungsdarstellung der verwendeten Größen zu finden.
- Die Auswahl geeigneter Datenstrukturen für die verwendeten Größen ist fast immer ein wesentlicher Schritt der Programmentwicklung. Das Rad nicht neu erfinden: Kenntnis von Standard-Datenstrukturen ist hier für den Erfolg sehr wichtig!

Auswahl/Entwicklung von Algorithmen

Mit diesen Daten soll natürlich auch etwas passieren: aus der Aufgabenstellung ist ein Berechnungsverfahren zu gewinnen, ein Algorithmus.
Hierbei sollte man zwar schon an die folgende Implementierungsarbeit denken, aber sich noch nicht mit zu vielen Details (etwa der verwendeten Programmiersprache) befassen.

Implementierung

Ist man sich über Algorithmen und Datenstrukturen im Klaren, muss das ganze in ein Programm(-system) umgesetzt werden, das unser Rechner verarbeiten kann.
Wichtige Frage dabei: was für Werkzeuge gibt es bereits, die uns die Arbeit erleichtern?

- Unumgänglich für praktisch jede Programmentwicklung: Übersetzer einer Hochsprache.
  Die Programmiersprache, die wir in dieser Vorlesung benutzen werden, heißt Ada 95
- Gibt es fertige Programmmodule, die uns unterstützen, z.B.: sollen wir zum Abspeichern unserer Daten ein Datenbanksystem kaufen oder programmieren wir alles selber? Auch hier gilt mal wieder: nicht das Rad neu erfinden! Im Netz sind umfangreiche Programmsammlungen verfügbar, zum Teil (nicht immer!) in sehr guter Qualität.

Implementierung

- Im Beispiel Wettervorhersage sollten wir uns die vorhandenen Programmsysteme zum Lösen von Differentialgleichungen ansehen, auf jeden Fall die vorhandenen Numerikbibliotheken sichten; ziemlich sicher werden wir ein Programm zur Visualisierung dazukaufen.
- Bei einem komplexen Vorhaben wird auch Software nützlich sein, die die Programmentwicklung unterstützt (für unsere Übungsaufgaben wird das keine Rolle spielen): Dokumentation der Programmentwicklung, Unterstützung von Teamarbeit etc. sind Aufgaben, die man nicht unterschätzen sollte.

Was sonst noch zum Entwicklungsprozess dazugehört

- Analyse des Produktes: Arbeitet es korrekt? Arbeitet es effizient? Kritisches Hinterfragen während aller Entwicklungsschritte!
  Die Formalisierung ermöglicht es, präzise Aussagen dazu zu machen. (Ob die Formalisierung selber korrekt war, ist ein anderes Problem).
- Darstellung und Interpretation der Ergebnisse
- Wartung und Weiterentwicklung
- ...

Wie geht es weiter?

- Spielerische Einführung in Ada 95 mit dem System AdaLogo `http://www.adalogo.de.vu/`
  - Grundlegende Konzepte der Programmierung
  - Realisierung in AdaLogo und Ada 95
  - Formalisierung und theoretischer Hintergrund
- Einfache Datentypen (Dinge, mit denen ein Computer typischerweise arbeitet: Zahlen, Buchstaben . . .) und Ausdrücke
- Kontrollstrukturen: Fallunterscheidungen, Schleifen
- Prozeduren und Funktionen