Grundlagen der Informatik für Ingenieure I

1.1 Hörerkreis und Quiz

Für welche Studienrichtungen ist diese Vorlesung geeignet?:

WW, MB, W-Ings (jeweils neue PO), CIW (auf freiwilliger Basis), sonstige?

Nicht geeignet (wg. Prüfungsord.) für: E-Technik, Mechatronik

Wer hat schon einmal ein Programm mit mehr als 50 Zeilen Code geschrieben?

Ja: Nein:

Wenn ja, welche Sprache?

Pascal: Fortran: C: C++: sonstige:

Wie schnell schreiben Sie auf einer Tastatur?

pro Minute: 20 Anschläge: 40: mehr als 40:

Wieviel Zeit haben Sie für praktische Übungen am Rechner eingeplant?

pro Woche: 2 h: 4h: 6h: >6h

Wer hat sich von Ihnen schon einmal Informationen über das Internet beschafft?

Ja: Nein:

1.2 Struktur der Lehrveranstaltung

- Vorbemerkung:
  - Es gibt heute kaum noch Gebiete in den Ingenieurwissenschaften, in denen nicht **auch** fundiertes Informatikwissen unabdingbare Voraussetzung für eine erfolgreiche und zukunftsorientierte Forschung, Lehre und Produktentwicklung ist.
  - Schon heute liegt die "Elektronik-/Informatik - Wertschöpfung" bei einem modernen KFZ bei Werten bis zu 30%; mit einem Prozentsatz von über 50% wird zwischen 2005 und 2010 gerechnet (Drive by Wire).
  - Die Vorlesung mit Übungen Grundlagen der Informatik (GdI) 6h (für MB 7h) kann vom Umfang her nur eine einführende Lehrveranstaltung sein. Die 6 SWS entsprechen etwa 4% Ihrer Gesamtstundenzahl im Studium. Schlußfolgerungen?

1.2 Struktur der Lehrveranstaltung

- Zum Inhalt:
  - Das Grundkonzept der Vorlesungen “Grundlagen der Informatik für Ingenieure” beruht darauf, den Hörern die Fähigkeit zu vermitteln, Problemstellungen der Informatik im Umfeld technischer Systeme zu lösen. Dabei wird, quasi als “roter Faden”, die Programmierung technischer Systeme als zentrales Thema behandelt, wobei immer dann, wenn es der Stoff erfordert, Kenntnisse aus den Fachgebieten Rechnerarchitektur, Betriebssysteme, Verteilte Systeme, Rechnernetze, Compilertechnik, usw. vermittelt werden.
  - Gliederung der Vorlesung (Reihenfolgen können sich ändern)
    - Einführung und Informationen zur Vorlesung
    - Einführung in Java (JDK)
    - Umgang mit UNIX
    - Java-Sprachkonstrukte
    - Klassen, Objekte, Methoden
    - Java-Applets

1.2 Struktur der Lehrveranstaltung

- Gliederung der Vorlesung (cont):
  - Einfache Graphik
  - AWT Abstract Windows Toolkit Teil 1
  - Programming in the Large/Small
  - Interaktivität und Ereignisbehandlung (Eventhandling)
  - Programm- und Datenstrukturen
  - Ausnahmenbehandlung, Fehlersituationen (Exceptionhandling)
  - Streams, I/O (Java-Ein-/Ausgabesystem)
  - Aktivitätsträger (Prozesse/Threads)
  - Images, Animation, Sound
  - Verteilte und Parallele Programmsysteme (Einführung)
  - AWT Abstract Windows Toolkit Teil 2
  - Programmiersprachen im Vergleich
  - Ausblick und Repetitorium

1.3 Überblick Programmiersprachen

- gegliedert nach Anwendungsgebieten
- Maschinenorientierte Programmiersprachen
  - Prozessorspezifische Assembler Sprachen
- Problemorientierte (höhere) Programmiersprachen
  - Technisch/Wissenschaftlich
    - C (C++) - insbesondere Systemprogrammierung
    - Fortran90/95 - insbesondere numerische Anwendungen
    - Java - insbesondere für Client/Server-Applikationen in Netzwerken und für grafische Oberflächen, *embedded systems,* architekturunabhängige Softwaresysteme
    - Algol, Pascal, Modula - kommerziell eher unbedeutend
  - Betriebswirtschaftlich orientierte Sprachen
    - Cobol,..
  - Datenbanken
    - SQL, Natural,..

1.3 Überblick Programmiersprachen

- Echtzeitsysteme (Realtime)
  - PEARL, ADA, ..
- Wissensbasierte Systeme (KI)
  - Lisp, Prolog

◆ Makrosprachen
  - PERL, AWK,..

◆ “Dokumentbeschreibungs” - Sprachen
  - Postscript, HTML, ..

■ andere Unterscheidungskriterien
  - objektorientiert/objektbasiert (C++, Smaltalk, EIFEL, JAVA)
  - funktional/prozedural/deklarativ
  - direkte Abarbeitung/interpretative Abarbeitung

1.4 Weiteres zur Lehrveranstaltung

■ Zur Vorlesung(allgemein):
  - einführende Vorlesung mit Hintergrundinformation (Rechnerarchitektur, Betriebssysteme, Laufzeitsysteme)
  - Rückkopplung wichtig; Zwischenfragen sind erwünscht.
  - Fehler auf den Folien sind nicht auszuschließen; die Beispiele sind nicht immer “compiliert”! Seien Sie also kritisch!

■ Einführende Veranstaltung meint
  - es wird grundsätzlich nichts Außergewöhnliches vorausgesetzt
  - sollte ein “Programmier(halb)wissen” vorhanden sein - verdrängen Sie es; versuchen Sie nicht in einer Ihnen evtl. schon bekannten Sprache zu denken und dann eine Übersetzung nach JAVA vorzunehmen.

1.4 Weiteres zur Lehrveranstaltung

- objektorientiert, warum?
  - Die objektorientierte Programmierung geht in ihren Anfängen auf die 2. Hälfte der 60er Jahre zurück (SIMULA) und ist damit - insbesondere für Informatiker - nichts wirklich Neues. Obwohl diese Vorgehensweise eigentlich einer “natürlicheren” Denkweise entspricht (was noch demonstriert werden soll), beginnt sie sich erst in letzter Zeit allmählich durchzusetzen.
    Ein Grund dafür ist die Tatsache, dass diese programmiertechnische Vorgehensweise zu geringerer Laufzeiteffizienz führt und der für dieses Programmiermodell notwendige “Unterbau” in früheren Zeiten nur rudimentär vorhanden war. Das Dilemma Programmstruktur versus Laufzeiteffizienz wurde in der Vergangenheit meistens zugunsten der Laufzeiteffizienz entschieden, mit allen Nachteilen wie z. B.:
    - schlechte Wartbarkeit des Codes,
    - schwierige Schnittstellenfestlegungen im Rahmen von Aufgaben, die ein Einzelner nicht mehr erledigen kann,
    - kaum Methoden, die die Einhaltung von Schnittstellen erzwingen.

1.4 Weiteres zur Lehrveranstaltung

- objektorientiert, warum?(cont)
  - Inzwischen beginnt sich aufgrund der modernen leistungsfähigen Prozessoren die Auffassung durchzusetzen, einen Teil des Zugewinns an Rechenleistung zugunsten besserer Programmstrukturen zu investieren. Eine Ausnahme bilden Applikationen aus dem Bereich des numerischen Rechnens auf Hochleistungsrechnern.
- Was gewinnt man mit dem objektorientierten Ansatz?
  - ◆ Das wird im laufenden Semester die Vorlesung zeigen; vorab einige Schlagworte:
    - wohldefinierte Schnittstellen
    - gekappselte Module, Baukastenkomponenten
    - Wiederverwendbarkeit (ohne Kenntnis der Implementierung)
    - einfachere Softwarewartung und -erweiterung
    - Beherrschbarkeit komplexer Systeme
    - Verteilte Programmsysteme

1.4 Weiteres zur Lehrveranstaltung

- JAVA, warum?
  - Java ist eine “Weiterentwicklung” von C++(ebenfalls eine objektorientierte Sprache) ursprünglich mit dem Ziel, eine möglichst einfache Sprache für den sogenannte “embedded” Bereich zu schaffen (von der Roboter-steuerung bis hin zur Waschmaschine oder zum Mobiltelefon).
  - Man entschloss sich, das Problem “Kompatibilität”[1] dadurch zu lösen, dass man mittels einer hardwareabhängigen Softwareschicht eine einheitliche “Abstrakte Maschine” definiert. Diese Maschine arbeitet den Java-Programmcode (Bytecode) interpretativ ab. Für jeden Hardwaretyp ist also nur einmal der Aufwand der Implementierung der Java-Maschine zu treiben, anstatt für jede Applikation x Varianten für jede potentielle Hardwareplattform vorhalten zu müssen.
    - Diese Tatsache führte zu dem Werbeslogan von Sun Microsystems

      Write once - Run everywhere

1. Ein Programm ist unabhängig von der zugrunde liegenden Hardware ablauffähig

1.4 Weiteres zur Lehrveranstaltung

- JAVA, warum?(cont)
  - Der objektorientierte Ansatz ist insbesondere auch für verteilte Anwendungen geeignet. So stellte sich sehr bald die besondere Eignung von Java zur Lösung von Aufgabenstellungen im Internetbereich heraus.
  - Darüberhinaus versprechen die Security-Eigenschaften von Java, die Lösung der Sicherheitsprobleme im Internet wesentlich voranzubringen.
  - Last but not least
    Wie so oft in den vergangenen Jahrzehnten kommt es darauf an, mit einer (nicht immer unbedingt neuen (oder gar guten)) Idee zur rechten Zeit am Markt zu sein, mit dem finanziellen Background diese Idee zu “pushen” und den Mitstreitern schmackhaft zu machen. Dies ist Sun offensichtlich gelungen - wobei im Gegensatz zu Microsoft - Sun das Bestreben hat, sich in einem “offenen” Markt dem Wettbewerb zu stellen.

1.4 Weiteres zur Lehrveranstaltung

◆ Wie Sie noch sehen werden, ist die verwendete Programmiersprache eher zweitrangig - wir werden für den eigentlichen Sprachumfang höchstens 20% der Zeit investieren - wichtig ist das Erlernen der Programmiertechnik als Handwerkszeug. Darüberhinaus sind die grundlegenden Sprachelemente in allen Sprachen mehr oder weniger ähnlich.
Beherrscht man also grundsätzlich die “Kunst des Programmierens”, so sollte die Wahl der Sprache eher aufgrund anwendungsspezifischer Gegebenheiten vorgenommen werden.

1.5 Zu den Übungen

■ Zu den Übungen:

- Leider stehen zur Betreuung neben den Dozenten nur 2 (??) Stud. HKs zur Verfügung (dies ist kein Geldproblem!)
- 2h betreute Übung; d.h. Sprechstunde/Fragestunde vor Ort am Objekt
- 1h Beratung in der Sprechstunde (Dr. Wurm); “abseits” der Übungen
- freies Üben; Learning by Doing! Programmieren kann man nicht “lesend” lernen! (insgesamt halte ich mindestens 6h/Woche für erforderlich)
- Lassen Sie sich nicht verleiten, die Zügel schleifen zu lassen!
  Einzelaussagen sind fast immer trivial (besser: leicht einsichtig).
  Die Komplexität entsteht durch die (Un-)Menge von Details und deren (oft unvermuteten - weil vergessenen) Zusammenhänge
- Fehler, die man selbst einmal gemacht hat, werden zukünftig leichter vermieden!

1.5 Zu den Übungen

- Übungsablauf
  - Kleinere bis mittlere, z. T. in vorgegebene Rahmenprogramme einzubettende Aufgaben.
  - Kommen Sie bitte vorbereitet zur Übung; es muß Ihnen vor der Aufnahme der Arbeiten am Rechner klar sein, wie die Daten und die Programmteile zu strukturieren sind.
  - 3 bis 4 größere Hausaufgaben zum Abgeben.
  - Erlaubt sind Zweiergruppen.
  - Lösung und Abgabe der Hausaufgaben sind Voraussetzung für die Zulassung zur (Schein-) Abschlußprüfung.

1.6 Abschlußprüfung

- für diejenigen, die “nur” einen Schein brauchen
  - Praktische Aufgabe am Rechner zu lösen (3h) und eine ca. 20-minütige mündliche Prüfung
  - wer einen benoteten Schein braucht, sollte das **vor** der Prüfung kundtun!
- für diejenigen, die eine “studienbegleitende” Prüfungsklausur schreiben müssen
  WW, MB, W-Ings (jeweils neue PO)
  - Lösung einer praktische Aufgabe(3h) für den Schein. Der Schein ist Voraussetzung für die Zulassung zur Prüfungsklausur.
  - 90 Minuten Klausur (Vordiplom-Prüfung)

1.7 Anmeldungen

- Wer's noch nicht getan hat:
  Bitte melden Sie sich im Cip-Terminalraum 1. Stock Inf.-HH unter dem Loginnamen: "gdijava" bis spätestens Freitag (19.04.) an.

1.8 Vorlesungsunterlagen/ Empfohlene Lehrbücher/andere Quellen

- Vorlesungsunterlagen/Empfohlene Lehrbücher/andere Quellen
  - Vortragsfolien werden im Internet spätestens 1 Woche vor ihrer Behandlung zur Verfügung gestellt.
  - Java - Bücher gibt es wie Sand am Meer!
    Die Vorlesung orientiert sich zum Teil an:
    - Teach Yourself Java 2 in 21 Days, Laura Lemay u. a.,
      Sams-Publishing, ISBN: 0-672-31438-X
    - The Java-Tutorial
      http: //www4.informatik.uni-erlangen.de/Services/Doc/Java/tutorial
    - auch als Buch erhältlich: The Java-Tutorial, JavaSoft
      Addison-Wesley-Verlag
  - Sprachdefinition und Informationen zur aktuellen Entwicklung finden Sie unter:
    - http: //www4.informatik.uni-erlangen.de/Services/Doc/Java

1.9 Termine

- Termine:
  Vorlesung H8: Dienstag 14:15 bis 15:45
  H9: Mittwoch 14:15 bis 15:45 (ggf. 15:45)
  Einer der beiden Termine 14-tägig, welcher ??
-
- Übungen: 02.151; Sparc-Ultra10 - Cluster
  - betreute: Dienstag: 10:00 - 12:00
    Mittwoch: 12:00 - 14:00
    Donnerstag: 8:30 - 10:00, 10:00 - 12:00
    Freitag: 12:00 - 14:00
  - freie: jederzeit,
    soweit Terminals verfügbar. In den mit GdI gekennzeichneten Zeiten an der Tür des Terminalraums haben Sie Vorrang; Sie können also ggf. jemanden “freundlich!!” verdrängen. Sollte das nicht funktionieren, wenden Sie sich bitte an uns!

Notizen