program Einführung in die EDV (input, SCHNEIDER, output, Studenten);

```
const erster Studienabschnitt in ET: real;
type Schneider: array (1..maxint,1..maxint,1..maxint) of ???;
var Hörerzahl: integer,
    Vorlesungsinhalt: Schneider,
    Erfolgsquote: not integer,
    Scripten: einzelne blätter z.T.großzügig raumgreifend handge-
              schrieben, trotzdem ziemlich unübersichtlich, ohne
              Inhaltsverzeichnis aber sehr aufwendig numeriert von
              A 1 - 1, bis C N-M;

PROCEDURE Vorlesung (var A,B,C:text);
BEGIN
```

Es trifft wohl am besten zu, wenn ich sage, ich blickte dieser Vorlesung mit sehr gemischten Gefühlen und Erwartungen entgegen. Einerseits kommt ein Techniker heutzutage um eine Auseinandersetzung mit diesen beeindruckenden Maschinen nicht herum; diese Vorlesung ist daher einfach eine Notwendigkeit. Prof. Schneider steht außerdem im Ruf, einen interessanten und guten Vortrag zu haben, und interessiert hat es mich auch - was ich nicht gerade von jeder Vorlesung behaupten kann.

Aber andererseits haben Prüfungen bei Prof. Schneider auch den Ruf, zu den Schwierigsten dieses Studiums zu gehören; Reglungstechnik gehört wohl zu den größten Hürden des 2. Studienabschnittes.

Die Vorlesung setzt sich inhaltsmäßig aus drei Teilen (A,B,C) zusammen, die zum Teil nochmals unterteilt werden:
Differentialquotient, Integrationsregeln, Differentialgleichungen, Reihen, Numerische Verfahren, Einschwingungsvorgänge und die mechanischen Analogien, Pascal, eine sehr kurze Fassung des Microsoft Diskettenorganisationssystems, Digital Rechentechnik, Binär-Code, Boolsche Algebra, Schaltfunktionen und deren technische Realisierung, Karnaught-Veitsch Diagramm, Microcomputertechnik, Assembler-Code etc.

Ein sehr vielfältiges Programm - und alles in einer zwei!-stündigen Vorlesung im ersten Semester. Bitte sehr, das ist doch keine EINFÜHRUNG IN DIE EDV. Das sind doch zu einem nicht unerheblichen Teil Dinge, die in Mechanik, in Mathematik 1 - 4, in Wechselstromtechnik, in Elektronik, in Microcomputertechnik und anderen Vorlesungen zum Teil im zweiten Studienabschnitt gelesen werden. Es gibt nichts gegen Pascal einzuwenden. Aber wenn ich mir Pacal-Lehrbücher ansehe, so beginnen alle mit den gleichen einfachen Programmbeispielen: Zahlen ordnen, Buchstaben sortieren oder so ähnlich. Da kommen keine Differentialgleichungen vor - wozu auch, das lernt man ohnehin in Mathematik. Auch von E-Technik braucht man nur so viel zu verstehen, daß man den Rechner einschalten kann. Was ein Kondensator ist, hat für die Erstellung eines Programmes keine Bedeutung.

Aber wo war die Einführung in die EDV geblieben?
Darunter hatte ich mir eigentlich vorgestellt, daß ich nicht wie der Ochs vorm neuen Tor stehe, wenn ich vor einem Computer sitze - auch wenns kein Sirius ist; daß ich über die Möglichkeiten und Unterschiede der verschiedenen Programmiersprachen einigermaßen Bescheid weiß;
Daß es zwischen den verschiedenen Systemen innerhalb der gleichen Sprache Unterschiede gibt und worauf man zu achten hat, wenn ein Programm auf verschiedenen Systemen laufen soll, daß ich grundsätzliche Dinge über Computer Programmiertechnik erfahre, wie zum Beispiel:was ist ein Top-Down-Entwurf? Was ist Pseudo-Code und wozu ist er gut? Wie konzipiert man ein längeres Programm, so daß man sich in der zweitausendsten Zeile noch auskennt? Warum soll man nicht am Bildschirm programmieren? Was muß ich über das System wissen, um Software dafür schreiben zu können. Was steht in der Dokumentation über einen Computer? Wie stelle ich fest, ob ein Algorithmus schneller als der andere ist? Wie organisiere ich die Speicherplatzbelegung möglichst sparsam, daß ich Grundsätzliches über Programm und Datenstrukturen erfahre. Pascal ist die am besten strukturierte Sprache; aber das nützt mir nichts, wenn ich es nicht ausnützen kann.
Daß ich etwas mehr über das Betriebssystem erfahre: Was ist die Peripherie und wie spreche ich sie an? Was gibt es für Speichermöglichkeiten? Was ist ein Winchesterlaufwerk? Welche Drucker gibt es und wie schnell sind sie? Was ist meta-language? Wozu sind Metacommands? Was ist ein 2-Schritt Compiler? Was ist ein Batchfile? Wozu ist er gut, wie legt man ihn an? Was sind predeclared functions?

Viele Fragen - wenige Antworten

Ich bin übrigens inzwischen selbst draufgekommen, was ein Batchfile ist. Es wäre mir aber lieber gewesen, wenn's mir jemand gleich erklärt hätte - z.B. statt dem Runge-Kutta Verfahren. Das steht nämlich in jedem Mathematik-Buch und lernen tut man es auch noch ein paar Mal.

Es ist völlig logisch, daß Vorlesungen in höheren Semestern als dem ersten auf Teile des Stoffes früherer Semester aufbauen. Umgekehrt finde ich es eigentlich nicht logisch, daß Bruchstücke des Stoffes kommender Vorlesungen im ersten Semester vorweggenommen werden. Wozu soll das gut sein? Ich sehe den Grund nicht.

Was noch hinzu kommt: Das kann sich kein Mensch mit Mittelschulbildung merken. Einfach weil lernen eben nur in einem bestimmten Zusammenhang sinnvoll ist. Die mathematische Beschreibung des elektromechanischen Feder-MasseReibungs Analogons baut auf dem Verständnis von physikalischen Vorgängen und mathematischen Verfahren auf, die auch kein HTL-Absolvent hat. So etwas kann man nicht in einer halben Stunde verstehen.
END;

PROCEDURE Hörsaalübungen (var an die Wand projezierte Programme: text);
BEGIN
Es werden Hausaufgaben verteilt und Programme erklärt.
Auch etwas, daß ich nicht verstehe. Einführung in die EDV besteht aus zwei Stunden Vorlesung und einer Übungsstunde. Dieses Verhältnis ist schon einmal verkehrt herum. Fortran für Maschinenbau,zwei, semesterweise alternierende Vorlesungen (das halte ich für sehr vernünftig), heißt 2V+3Ü und spielt sich dann so ab, daß man solange Vorlesung hält, bis es sinnvoll ist, sich an den Rechner zu setzen. Dann übt man ein bißchen Betriebssystem und Programme Eintippen und dann hört man wieder Fortran im Hörsaal. Auch vernünftig. Nicht so bei der Einführung in die EDV; hier wird die ohnehin viel zu karg bemessene Rechenzeit im Hörsal damit vergeudet, Programme zu besprechen, die für jemanden ohne jede praktische Erfahrung völlig undurchschaubar sind. Als Pikanterie am Rande: Die Programmbeispiele in den Scripten bzw. auf der Leinwand sind zum Teil mit der Hand geschrieben. Wie jeder weiß, macht auch der beste Kopierer immer wieder Fehler,z.B. schwarze Punkte, wo am Original keine waren (wer ein wenig von Pascal versteht, weiß, was ein Punkt mehr oder weniger in dieser Sprache bedeutet; wer nichts davon versteht, dem sage ich es: soviel wie ein ganzes Wort in einer anderen Sprache) oder weiße Flecken, wo vorher vielleicht ein Beistrich war. Das macht einem ganz schön zu schaffen, wenn man nun nicht genau weiß: Gehört da jetzt ein ',' oder ein';', wo man doch weiß, wie teuflisch gefährlich die Satzzeichen in Pascal sind.
END;

PROCEDURE Rechnerübung (var erste, zweite, dritte: selber üben);
BEGIN

Vor der Übung habe ich mir - wie 'befohlen' - eine Diskette um 90 öS gekauft. Bei einer Fortranvorlesung war das so: Es gab dieselben Disketten zu kaufen, ca 10 öS billiger, denn der Betreuer meinte:

'Ihr könnt sie auch selber kaufen, aber ich kriege sie billiger, weil ich ja mehr auf einmal kaufe. Aber wenn jemand überhaupt keine haben will, so kann ich ihm eine leihen.'
Wäre auch eine Möglichkeit, oder?
Später kommt man dann auch drauf, daß ein Sirius mit einem abgeschliffenen Schreib/Lesekopf bezüglich Diskettenqualität ganz schön anspruchsvoll ist.

Erste Rechnerübung - zu zweit an einem Rechner - zwei Stunden Zeit: Mein Kollege brauchte leider eine halbe Stunde, bis er seine persönlichen Daten eingetippt hatte (jeder kann halt nicht Maschineschreiben). Dann haben wir noch gemeinsam irgend ein Programm abgetippt; vor dem ersten Testen war jedenfalls die Zeit um.
Nachher habe ich mich gefragt, wozu ich dorthin gegangen bin?
Bei den nächsten beiden Übungsterminen war es nicht viel anders. Ich habe jedenfalls in einer 3/4 Stunde kein Programm zum Laufen gebracht. Meine Hoffnungen richteten sich auf die freien Rechentermine ab Jänner, wo man dann etwas länger üben könnte - so wurde gesagt: Das war aber nur die ersten beiden Male so; dann bin ich noch vier Mal umsonst in die Inffeldgasse gefahren. 12 Rechner für ungefähr 600 Studenten ist ja auch etwas zu wenig. Trotzdem ist es mir gelungen, mehrere Programme zu schreiben, allerdings: jetzt hätte ich gerne einen Betreuer gehabt, um ihn zu fragen, wo die Würmer in meinen Programmen sind; jetzt hätte ich wesentlich gescheitere Fragen als bei der ersten Rechnerübung stellen können - jetzt war aber keiner da und die Kollegen waren auch nicht wesentlich schlauer als ich.
Während der Ferien mußte man sich bei den 4 Rechnern in der Steyrergasse schon um 7.30 Uhr anstellen. Logisch - gleich nach den Ferien war die vierte Rechnerübung angesetzt - und die ist bekanntlich eine Prüfung.
END;

PROCEDURE vierte-übung (var ein Beispiel:Programm);
BEGIN

50 Minuten zum Programmieren, 10 Minuten zum Fragen beantworten;

20 Minuten brauchte ich, um das Programm einzutippen, 2 Minuten zum Testen, 5 Minuten Syntaxfehler ausbessern, 2 Minuten Testen, no errors, 3 Minuten Compilieren und Linken, Programm läuft nicht, Fehler im Algorithmus, 3 Minuten Fehlersuchen, 3 Minuten Compilieren und Linken, läuft wieder nicht, wieder Fehlersuchen, am Papier tue ich mir wesentlich leichter, aber diese grüne, unscharfe Schrift. Am Ausdruck würde ich den Fehler wesentlich schneller finden; am Bildschirm programmieren ist kein gutes System. Man hat keinen Überblick, man starrt oft minutenlang auf einen Fehler und sieht ihn einfach nicht. Nach 10 Minuten endlich: Brauche bloß zwei Variable zu vertauschen. 3 Minuten Compilieren und Linken. Programm läuft aber falsch. Zeit ist auch um. 5 Minuten nach der Prüfung hatte ich den Fehler gefunden. War bloß ein '-' zuviel.

Ist das wirklich sinnvoll, auf Zeit zu programmieren? Ich dachte, ein gutes Proramm ist ein Programm, das schnell läuft und wenig Speicherplatz braucht und nicht eines, das schnell geschrieben ist - die laufen oft langsamer. Bei den meisten Prüfungen kommt man auch durch, wenn man nicht nur für die Prüfung lernt, sondern einfach den Stoff gut kann. Hier ist es umgekehrt. Wenn man ein gut struktuiertes Programm schreibt, vielleicht Funktionen oder Proceduren verwendet, ist das für das Prüfungsergebnis sehr schlecht - weil man dann nämlich viel langsamer ist, als ein anderer, der einfach die Befehle von oben nach unten hinschreibt - wiewohl das bei kurzen Programmen natürlich geht. Aber ob's einen Sinn hat, steht auf einem anderen Blatt.

```
END;

PROCEDURE Prüfung (Noten);

BEGIN (* $DAS HAUPTPROGRAMM KÖNNT IHR SELBST SCHREIBEN!
         WIE UND WO IHR'S DANN LAUFEN LASSEN KÖNNT, IST
         JA BEKANNT *)

END.
```