Zeitschrift: Visionen : Magazin des Vereins der Informatik Studierenden an der ETH Zürich

Herausgeber: Verein der Informatik Studierenden an der ETH Zürich

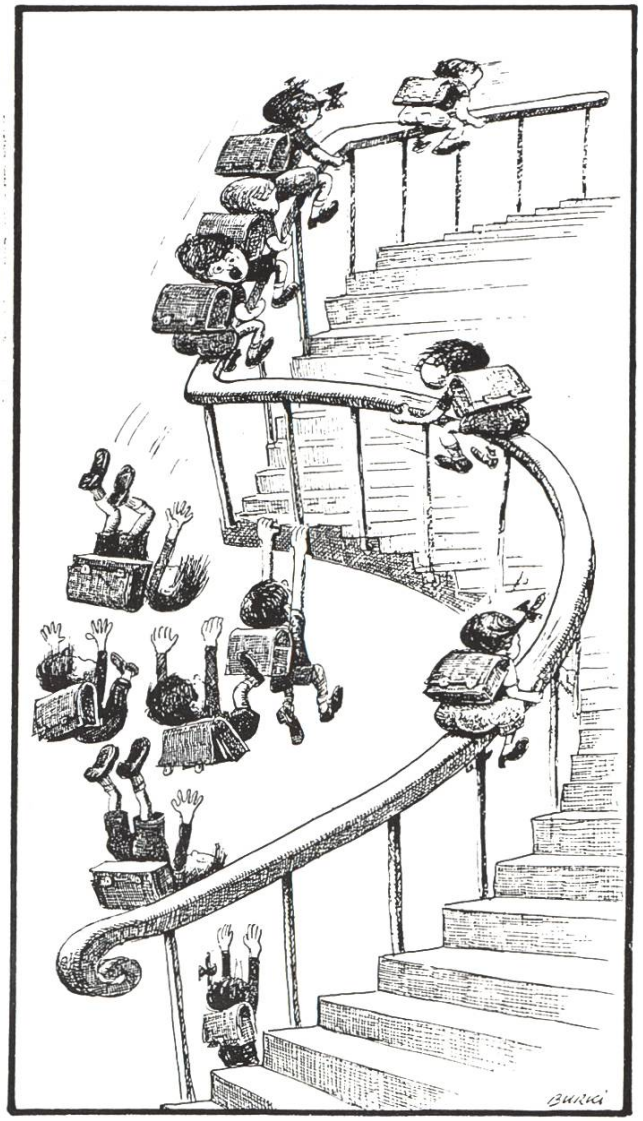
Band: - (1993)

Heft: 11

Heft

Visionen

11
November
93



vi

Prüfungsstatistiken
Programmierwettbewerb

VISIONEN Herausgegeben vom Verein der Informatikstudierenden an der ETH Zürich

Adressen

Aktuar: Stefan Rohmer
Keltenstrasse 6, 8044 Zürich
Tel. 01 / 251 34 51
e-mail: stefan@vis.inf.ethz.ch

Exkursionen: Boris Nordenström
Hardstrasse 324, 8005 Zürich
Tel. 01 / 273 24 80
e-mail: banorden@iiic.ethz.ch

Feste & Kultur: Frank Möhle
Dielsdorferstrasse 7, 8155 Niederhasli
Tel. 01 / 851 03 21
e-mail: fmoehle@iiic.ethz.ch

Präsidentin: Grete Danielsen
Dohlenweg 26, 8050 Zürich
Tel 01 / 302 48 97
e-mail: gcdaniel@iiic.ethz.ch

Quästor: Daniel Kluge
Irringersteig 3, 8006 Zürich
Tel. 01 / 252 04 14
e-mail: dankluge@iiic.ethz.ch

Redaktor: Patrick Leoni
Sandstr. 6, 8610 Uster
Tel. 01 / 940 05 14
e-mail: pleoni@iiic.ethz.ch

Verleger: Hans Domjan
Kapfhalde 3, 6020 Emmenbrücke
Tel. 041 / 53 68 83
e-mail: hdomjan@iiic.ethz.ch

Visinfo(Infosystem): Michel Müller
Rheinländerstr. 15, 4056 Basel
Tel. 061 / 321 81 23
e-mail: mmueller@iiic.ethz.ch

Prüfungen und Unterricht:
Leonhard Jaschke
Südstrasse 67, 8008 Zürich
Tel. 01 / 383 60 55
e-mail: ljaschke@iiic.ethz.ch

Impressum

Herausgeber:
Verein der Informatikstudierenden an der ETH Zürich.

Verleger: Hans Domjan
Redaktor: Patrick Leoni

Adresse Verlag & Redaktion:
VIS
Verein der Informatikstudierenden
Haldeneggsteig 4, IFW B29
ETH Zentrum
8092 Zürich

Tel: 01 632 72 12 (Mo-Fr, 12.15-13.00)
Fax: 01 262 39 73
e-mail: vis@iiic.ethz.ch
Postkonto 80-32779-3
Präsenzzeit: Mo-Fr: 12.15-13.00

Auflage: 1400
Inseratepreise:

1 Seite	s/w	SFr. 500.–
1 Seite	Farbe	SFr. 750.–
1/2 Seite	s/w	SFr. 250.–
1/4 Seite	s/w	SFr. 150.–

Redaktions- und Anzeigeschluss für die nächste Ausgabe:
Freitag, 3. Dezember 1993

Visionen

Hoi Zäme

Wie Ihr seht, habt ihr Euer HZ wieder im guten, alten Gewand zurück, was aber nicht bedeutet, dass ich hie und da auch wieder mal eine kleine Wikingermessage reinpacken werde.

Uns liegen jetzt wieder die neusten Zahlen über die Prüfungen vor. Auf den ersten Blick freut es uns natürlich sehr, dass die Durchfallquote des zweiten Vordiploms endlich mal tiefer ist als diejenige des ersten. Mit weit über dreissig Prozent behält unsere Abteilung dennoch ohne Probleme eine Spitzenposition. Diese Tatsache scheint auch langsam den Weg an die Kantonsschulen gefunden zu haben; wie sonst könnte man erklären, dass sich dieses Jahr lediglich noch 131 Studenten (weibliche Form nicht nötig...) eingeschrieben haben. Was die Resultate vom Schlussdiplom betrifft, verweise ich auf die Stellungnahme an anderer Stelle in diesem Heft.

Endlich konnte in diesem Jahr auch wieder der Programmierwettbewerb im Rahmen der Ausscheidung für den ACM International Collegiate Programming Contest (West European Regional) durch den VIS durchgeführt werden (der letzte hatte 1989 stattgefunden!!!). Sponsoren und freiwillige Helfer haben einen reibungslosen Ablauf ermöglicht - THANKS! Ich möchte den Siegern in Zürich nochmals recht herzlich gratulieren (Rangliste in diesem Heft). Wenn diese Visionen erscheinen sind unsere Teams auch bereits wieder aus Swansea zurück und Frank wird dann in der nächsten Ausgabe genauer über das Ganze berichten.

Ich hoffe auch, dass alle Erstsemestrigen sich jetzt gut eingelebt haben. Der Survival Guide ist auf jeden Fall inzwischen auch unter Professoren und Assistenten aus den verschiedensten Gründen zum absoluten Renner geworden. Das FEST war auch ein voller Erfolg und wir freuen uns bereits auf die nächsten zwei F&K-Anlässe:
FIGUGEGL:
Donnerstag 2. Dezember
ROCKY X-MAS:
Donnerstag 21. Dezember

Mit grossem Bedauern müssen wir schon wieder ein Vorstandsmitglied verabschieden: Maxim hat jahrelang für den VIS das VISINFO geführt. In seiner Hand wurder das VISINFO auf die neue Maschine portiert, auf der es jetzt läuft und in das ezInfo integriert. Mit ihm verlieren wir ein erfahrenes und wichtiges Mitglied im Vorstand. Wir vom Vorstand sind betrübt darüber, dass Maxim uns nicht mehr zur Verfügung stehen kann und wünschen ihm in seinem weiteren Schaffen viel Glück und Erfolg.

Ok - dere får ha det så bra til neste gang!

Grete

Freinacht

Jaja, derweil sich die einen an der Bar im *Hilton* zu Swansea (GB) vergnügen, die anderen sich im Kino *The Life of Brian* oder gar *Aladdin* 'reinziehen, wieder andere sich in den Armen des oder der Liebsten *'rumwälzen* und die ganz Tüchtigen bereits den Schlaf des oder der Gerechten *schlafen* sitze ich hier im VIS-Büro, allein, einsam, und *layoute* die Visionen. Naja, ganz allein bin ich ja nicht, doch was für eine Gesellschaft ist schon so ein *NeXT*. Nun, wie auch immer, ich habe den Job gewollt, jetzt steh' ich ihn auch durch, nicht wahr. Job? *Beruf!* Redaktor zu sein der schönsten Studentenzeitung weit und breit, Redaktor der Visionen, eine Traumbeschäftigung für einen verhinderten Journalisten wie mich...

Warum diese Zeilen? Einfach so, weil ich halt wieder einmal zu wenig Texte hatte? Im Gegenteil: Was Euch LeserInnen diesmal so geboten wird, ist von einer selten erreichten Vielfalt –: Nach dem obligatorischen *Hoi Zäme* geht es los mit dem mathematischen *Background* des letzten Visionen-Covergirls (das mit den Ljapunov-Kurven), einer Reaktion eines *Cray*-Fans auf den *HLR*-Verriss vom letzten Mal, der elften Folge von Chris Flu's famoser *Kochecke*, der sechsten von Masus' *Spielzeugen*, einem Bericht von der *Programmierparty* vom 7. November und einem saftigen Stück über die Durchfallquoten. Danach verhilft Euch der VIS zum *Computer-Guru*-Status (wer wollte das nicht schon immer sein?), resümiert die *Prüfungsversagerquoten* der letzten Jahre und gibt die neuesten Infos übers neue *Software-Copyright* durch. Dann erwartet Euch ein *Sakrileg*. Ein kleines zwar nur, aber immerhin. Ich verrate nur soviel: Dem *Elfenkönig* erwächst schwere Konkurrenz...

Also nochmal: Warum diese Zeilen? Nun, es ist nicht leicht zu sagen, aber ich brauche einen Nachfolger. Ab März werde ich *diplomarbeiten*, und mit einem hohen Amt im VIS, wie es ein Redaktorjob unzweifelhaft darstellt, geht das nicht so gut. Darum: Wenn Du, LeserIn, Dir vorstellen kannst, diese *Postille* (mit-) zu *managen*, so scheue Dich nicht, und komme zu uns. Die Sache hier ist zwar anspruchsvoll, aber echt interessant und lohnenswert. Und ich würde weiss was drum geben, die Visionen als Diplomarbeit anrechnen lassen zu können... aber das geht scheint's nicht.

Naja, vorerst (das heisst bis März) bin es ja noch ich, der die Visionen zusammenstiefelt, allein, einsam.

Doch der Starke ist am mächtigsten allein.

Habe ich einmal gelesen.

Euer Patrick

Entdeckungsreise im Ljapunovraum

Wie in den letzten Visionen angekündigt, möchte ich hier nun noch etwas detaillierter auf die Bilder eingehen, von denen ein Beispiel auf dem Titelbild der Visionen 9-10/93 zu sehen ist. Vor allem möchte ich zeigen, wie diese Bilder berechnet werden und nur am Rande auf die dahinterstehende Theorie eingehen.

Der Ljapunovexponent

Der Ljapunovexponent gibt an, wie "chaotisch" bzw. wie "dynamisch" eine Funktion ist. Genauer gesagt, er erlaubt, das *qualitative* Verhalten der Spur eines dynamischen Systems zu untersuchen. Für eine genaue Beschreibung des Ljapunovexponenten möchte ich auf entsprechende Literatur verweisen. Es interessiert hier nur, dass es im allgemeinen recht schwierig ist, den Ljapunovexponenten, in Zukunft λ genannt, zu berechen. Dies vor allem deshalb, weil λ nur in Ausnahmefällen in geschlossener Form dargestellt werden kann.
Der Ljapunovexponent ist definiert als

$$\lambda := \lim_{t\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \ln \left| \frac{df(x)}{dx} \right|_{x=x_t}$$

Bei der Berechnung von λ tritt das erste Problem schon beim Lim(t->∞) auf, da man auf einem Rechner nicht unendlich lange rechnen kann. Aus diesem Grunde muss hier nach einigen Iterationen die Berechnung abgebrochen werden.

Die dynamische Grundgleichung

Nun haben wir den Ljapunovexponenten definiert. Was uns also noch fehlt, um ein Bild zu berechnen, ist eine dynamische Funktion. Im folgenden wird also auch noch die *dynamische Grundgleichnung* betrachtet. Diese lautet

$$x_{t+1} = a x_t (1 - x_t)$$

Diese Gleichung kommt in vielen dynamischen Systemen vor. Sie beschreibt zum Beispiel in erster Näherung die Fischpopulation in einem Teich, in dem nur eine begrenzte Nahrungszufuhr existiert. Sind viele Fische im Teich, verhungern die meisten; die Population nimmt stark ab. Dadurch steigt das Nahrungsangebot wieder, und die Population nimmt zu. In Gleichung (2) kann nun die Anzahl Fische dem x(t) zugeordnet werden und die Nahrung dem Parameter a. Für gewisse a existiert nun eine eindeutige Lösung x(∞), für andere scheint die Funktion x(t) sehr dynamisch oder populär ausgedrückt: *chaotisch*. Es sei nun dem Leser überlassen, die Anzahl der Lösungen der Gleichung für bestimmte Parameter zu bestimmen.
Was uns nun an dieser Stelle inter-

essiert, ist der Ljapunovexponent der Gleichung (2). Setzen wir nun die dynamische Grundgleichung in die Definitionsgleichung des Ljapunovexponenten ein, erhalten wir:

$$\lambda = \lim_{t\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \ln \mid a - 2ax \mid_{x=x_t}$$

Um λ nun auf dem Computer zu berechnen, ersetzen wir die unendliche Summe durch eine endliche Summe, die aber ausreicht, um eine genügende Genauigkeit zu erreichen. Konkret schreiben wir:

$$\lambda = \frac{1}{n} \sum_{t=0}^{n-1} \ln \mid a - 2ax \mid_{x=x_t}$$

wobei x natürlich die n-te Iteration der dynamischen Grundgleichung (2) ist. Programmtechnisch muss also in einer Schleife die Summe berechnet und gleichzeitig die dynamische Grundgleichung iteriert werden.
Um nun das Bild auf dem Computer nun auch noch zweidimensional darstellen kann, wir aber nur eine Variable zur Verfügung haben, gibt es einen kleinen Trick: Wir benutzen an Stelle des einen Parameters x zwei Parameter ar und br. Mit diesen iterieren wir nun in einer weiteren Schleife über das ganze Bild. Um die Bilder nun noch interessanter zu gestalten, benutzen wir als Parameter a die Werte der Koordinaten ar und br. Diese werden abwechselnd benutzt, oder in irgend einer beliebigen Sequenz. Die abgebildeten Ausschnitte liegen alle im mehr oder weniger gleichen Ausschnitt, aber unterscheiden sich nur in der Reihenfolge, in welcher die Parameter ar und br benutzt werden. Aber bevor ich das hier länger mit Worten zu erklären versuche, sei auf das Listing verwiesen.

Noch ein Wort zum Rechnen

Für alle, die mal selber solche Bilder berechnen wollen und die spannende Reise im Ljapunovraum selbst unternehmen wollen, hier noch einige Tips:

1. Um vernünftige Bilder zu erhalten, sind viele Interationen nötig, in der Grössenordnung von 10000. Dies heisst, dass der Rechenaufwand relativ gross wird.
2. Eine gute Bildgrösse ist 400 * 400 Pixel.
3. Für jedes Pixel und jede Iteration müssen ein Logarithmus, 4 Multiplikationen und ein paar Additionen berechnet werden. Bei einem Bild mit obiger Grösse muss man demnach etwa mit 1.6 Milliarden Logarithmen rechnen. Aus diesem Grund muss man sich genau überlegen, wie diese Rechnung am besten auf die Hardware abgebildet wird. In meinem Fall war die Lösung, dass ich die Iteration über das ganze Bild laufen liess, anstatt über jedes Pixel einzeln. Dies mag erstaunen, aber wenn man sich mal ein bisschen mit einer Programmiersprache auseinanndergesetzt hat, ist es eigentlich gar nicht mehr sooo schlimm.

An dieser Stelle noch eine Bitte an alle, die mal rechnen wollen: Legt bitte nicht die ganzen Studentencluster an der ETH lahm, im speziellen die SPARCs im HG G26 und die Rif-Raf-Maschinen im IFW.

Aus dem Lexikon (l)

binary tree (n): see binary tree and binary tree

Das Listing

```
   LOGO2 = ALOG(2.0)

      DO 50 A=0, ((WIDTH-1)*(HEIGHT-1))
         TOT(A) = 0.0
         X(A) = 0.5
         AR(A) = 2.75 + MOD(A,400) * INCA
         BR(A) = 2.75 + INT(A/400) * INCB
50    CONTINUE

      DO 300 N=0,IT
         DO 200 A=0, ((WIDTH-1)*(HEIGHT-1))
            IF ( PARAM( MOD(N, 5)) .EQ. 0 ) THEN
               X(A) = AR(A) * X(A) * (1.0-X(A))
               TOT(A) = TOT(A) + ALOG(ABS(AR(A) -
               2.0*AR(A)*X(A)) + 1.0E-30) / LOGO2
            ELSE
               X(A) = BR(A) * X(A) * (1.0-X(A))
               TOT(A) = TOT(A) + ALOG(ABS(BR(A) -
               2.0*BR(A)*X(A)) + 1.0E-30) / LOGO2
            END IF
200      CONTINUE
300   CONTINUE

      DO 400 A=0,((WIDTH-1)*(HEIGHT-1))
         LJAP(A) = TOT(A) / FLOAT(IT)
         IF (LJAP(A) .GE. 0.0) THEN
            PIC(A) = 0
         ELSE
            PIC(A) = MOD(INT(-65535.0 * LJAP(A)), 65535) + 1
         END IF
400   CONTINUE
```

Chris Flu's Kochecke

Folge 11: Lauchcannelloni à la mode du chef

Die Ferien sind vorbei, die Prüfungen ebenfalls, die nächsten kommen bestimmt (wenn nicht tut es mir leid oder ich gratuliere), Zeit sich wieder den angenehmeren Seiten des Lebens zuzuwenden, nein, nicht dem Computer, sondern dem/der FreundIn (sofern man/frau nicht wegen den Prüfungen wieder einmal Single geworden ist). Beziehungen jeglicher Art wollen gepflegt sein und die Liebe geht bekanntlich durch den Magen und nicht durch den Scanner. Es ist dies nicht die erste Kochecke, welche sich den Berührungsängsten von InformatikerInnen mit dem anderen (oder gleichen[1]) Geschlecht widmet. Gerade jetzt, da sich meines Wissens ins erste Semester keine einzige (!!!) Frau eingeschrieben hat, scheint es wichtig, die Grundlagen für eine glückliche Zukunft der Erstsemestrigen durch Steigerung ihrer persönlichen Attraktivität zu schaffen. Wer nicht kochen kann muss fühlen. Kochen ist schmerzloser als ständige Abfuhren und Körbe, mal abgesehen von abgetrennten Gliedmassen und Verbrennungen dritten Grades, aber wer will deshalb gleich den 486er ins Korn werfen und Herzensdamen oder -herren in die Fünfsternspelunke ein laden, was üblicherweise den endgültigen Ruin zur Folge hat... also ich nicht! Nicht nur, dass ein gemütlicher Abend in den eigenen vier Wänden wesentlich günstiger kommt, lassen sich doch gerade bei sehr jungen Beziehungen eindeutige Rückschlüsse auf die charakterlichen Eigenschaften seines Gegenübers ziehen. Punkte die unbedingt beachtet werden müssen sind zum Beispiel: kriegt er/sie Hühnerhaut beim Anblick einer Dose (leih Dir zu diesem Zweck von Deinem Ururonkel eine solche Mühle aus), spielt er/sie gleich gut TETHris wie ich, kann er/sie abtrocknen und ist er/sie auch ohne Zwang gewillt, dies zu tun, mit welcher Baudrate arbeitet sein/ihr Modem. Bleibt noch die Frage, was denn gekocht werden soll. Der/die Kluge reist nicht nur im Zuge, er/sie liest vor allem Chris Flu's Kochecke, um sich kulinarisch auf dem laufenden zu halten. Während in den vergangenen Ausgaben vor allem monolithische Rezepte vorgestellt wurden, möchte ich hier erstmals ein unproblematisches Gericht im Client-Server-Modell präsentieren.

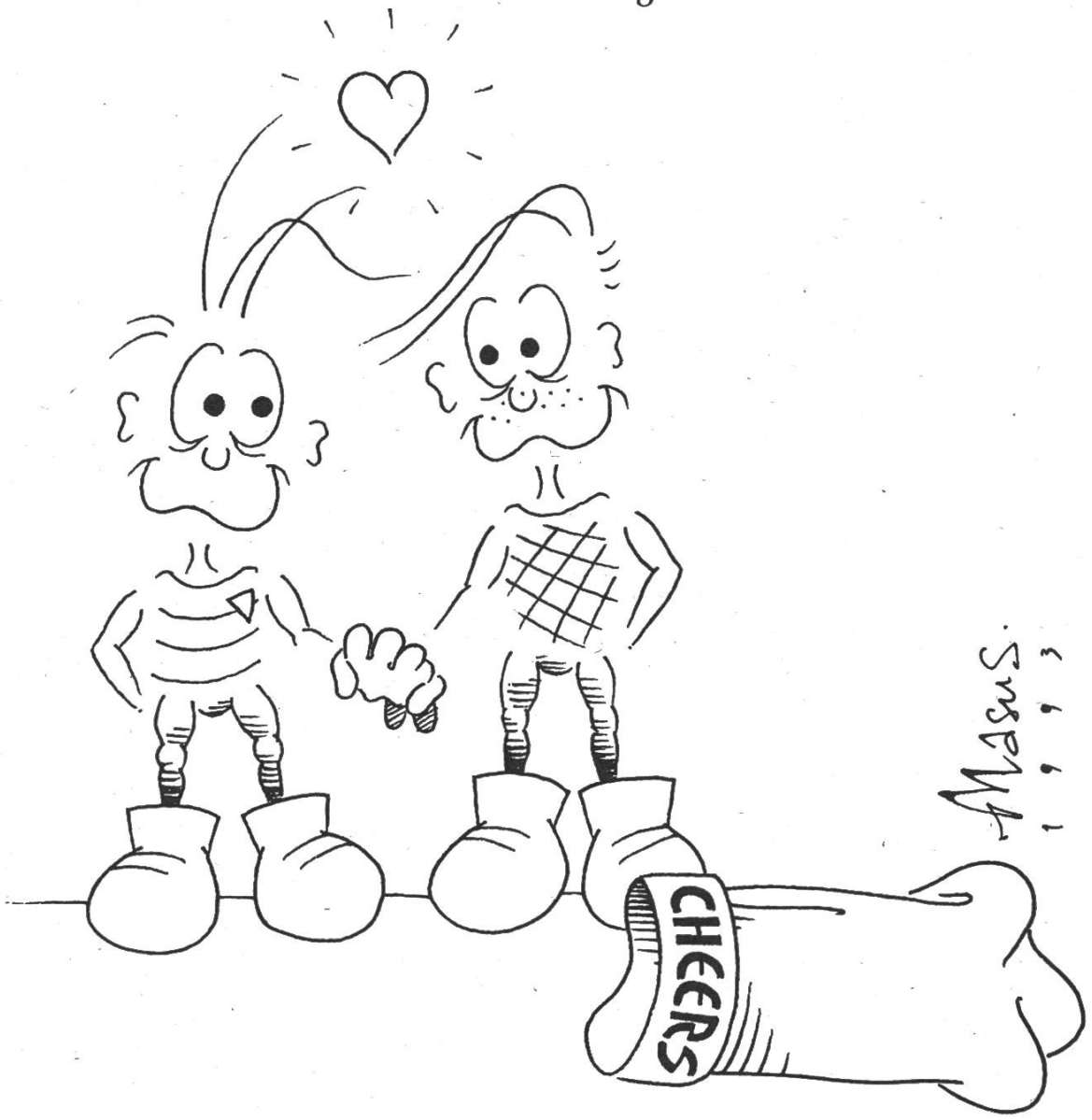
1 Jöö, das finde ich lieb! - Masus.

Man/frau benötigt (für 2-3 StudentInnen):

Cannelloni (1 Paket reicht sicher, die Menge ist ziemlich abhängig vom Rohrdurchmesser, es braucht soviele Röhren, bis die Füllung aufgebraucht ist, s.u.)

Füllung:
2 grosse Stangen Lauch
1 Zwiebel
1 Sack Sojabohnen-Keimlinge (250g)
200 g Feta oder anderer Schafkäse
1 Ei

Guss:
1 EL Butter
1 EL Mehl
3 dl Milch
1 Knoblauchzehe
120 g Sbrinz



Der Lauch wird gewaschen, in Scheiben geschnitten und mit der Zwiebel (gehackt), den Sojakeimlingen und wenig Oel so lange in der Bratpfanne gekocht (=gedünstet), bis nur noch wenig Flüssigkeit übrig bleibt. Dann wird die Masse vom Herd genommen und mit Ingwer, Sojasauce, Salz und Pfeffer gewürzt. Den Feta schneiden wir in kleine Würfel und mischen ihn sowie das Ei (ohne Schale) dazu. Eine Gratinform wird eingefettet. Die Cannelloni werden mit der Masse gefüllt und in die Form gelegt. Zum Füllen hält man die Cannelloni am besten auf der einen Seite mit der Hand zu und schiebt mit einem schmalen Teelöffel die Masse portionenweise hinein.

Die Butter lassen wir in einer Pfanne zergehen, geben das Mehl hinzu und rösten dieses so lange, bis es eine bräunliche Farbe angenommen hat. Alsdann giessen wir die Milch dazu. Das Rösten dauert nicht sehr lange, die Milch sollte deshalb bereits abgemessen sein, damit sie nur noch dazugeschüttet werden muss, wenn das Mehl bereit ist (das Mehl während dem Rösten in Bewegung halten (sonst wird es zu fett)). Mit dem Schwingbesen muss die Sauce kräftig umgerührt (nicht umgeschmissen...) werden, damit keine Knollen entstehen. Diese Grundsauce nennt sich Béchamelsauce und kann (entsprechend gewürzt) auch zu Teigwaren (Tortellini) gereicht werden. Dieselbige bringen wird nun unter stetem Rühren (damit sie nicht anbrennt) zum Kochen (dies wird *Aufkochen* genannt) und rühren die Knoblauchzehe (gepresst oder kleingeschnitten) und den Sbrinz (portionenweise, nicht alles aufs Mal) darunter – wer es ganz besonders gut machen will, fügt noch ein wenig Rahm bei. Dann giessen wir sie über die Cannelloni und backen (= gratinieren) diese im vorgeheizten Backofen bei 190°C während 30 bis 40 Minuten.

Der Tip: Das Gericht lässt sich bestens im voraus zubereiten, es muss dann nur noch gratiniert werden.

Auch diesmal wünsche ich Euch einen guten Appetit und viel Vergnügen beim Ausprobieren.

Chris Flu, IIIC/8

Aus dem Lexikon (II)

endless loop: see loop, endless

Dirty Harry kehrt zurück

Wie spätestetens seit dem IIIC Survival Guide (erhältlich im VIS-Büro) jedem klar ist, gleicht das Studium der Informatik an der ETH dem Überlebenskampf im prähistorischen Dschungel. Die Studis, *Jurassic Porks* gleich, schlagen sich möglichst unauffällig durchs Dickicht, immer auf der Hut vor den grossen Predatoren. Mit dieser Aufgabe haben sie es ja schon schwer genug.

Doch wer dann den Aufstieg zum *Rambosaurus Rex* geschafft (d.h. das zweite Vordiplom bestanden) hat, der sollte ja eigentlich einigermassen sicher sein.

Doch leider gibt es ein Raubtier, das manchmal auch Gefallen daran findet, einen Rambo zu reissen, immer nach dem Motto 'Du fragst Dich jetzt sicher, ob noch ein Punkt in der Kammer ist oder nicht...'.

Genug der grausigen Metaphern, ab jetzt wird's konkret:
Unserer Meinung nach kann man die hohen Durchfallquoten (die höchsten ETH-weit) im 1. und 2. Vordiplom gerade noch tolerieren, wenn man sie als 'notwendiges Sieb' für geeignete Diplomkandidaten betrachtet. Ein Studi, der beide Vordiplome erfolgreich bestanden hat, hat unserer Meinung nach bewiesen, dass er es wert ist, auch das Schlussdiplom zu erhalten.

In den Schlussdiplomprüfungen dann zeigt der Student, ob er sich mit dem Stoff auskennt und die richtige Denkweise beherrscht. Rein theoretisch kann ein Student hier schlecht abschneiden, doch nachdem die beiden Vordiplome absolviert wurden (als Sieb) sollte auf jeden Fall feststehen, dass das richtige Denken da ist.

Wenn also ein Professor in den Schlussdiplomprüfungen nicht mit den Leistungen eines Kandidaten oder einer Kandidatin zufrieden war, dann sollte er (unserer Meinung nach) eine 3.75 verleihen, mit der Botschaft 'Stoff nicht genügend bearbeitet.' Wenn die Prüfung wirklich unglücklich verlief, dann kann man immer noch eine 3.5 verteilen. Dies bedeutet dann 'Das war sehr schlecht!'
So halten es auch fast alle Dozenten. Fast alle.

Bis auf einen.

Mit erschreckender Regelmässigkeit verleiht Professor Mössenböck Noten unterhalb von 3.5. Uns ist völlig unverständlich, wieso dies geschieht. Bei einem Notengewicht von 25% ist es völlig klar, dass ein Kandidat mit

einer 2.5 als Note so gut wie durchgefallen ist, wenn er nicht woanders mindestens eine 5.5 erzielt.

Dies ist um so ärgerlicher, als bekanntlich Prof. Mössenböcks Prüfungen reine Wissensabfragen (z.B. 'Beschreibe das SCSI-Protokoll' oder 'Schreibe den Mark-And-Sweep-Algorithmus auf') sind. Seine Prüfungen messen also zu einem guten Teil die falsche Sache, nämlich die Fähigkeit, Fakten zu memorieren. Dies ist eine alte Tatsache, und als solche allen bekannt, die sich auf die Mössenböckschen Prüfungen vorbereiten. Aber: Auswendig lernen ist nicht jedermanns (oder -fraus) Sache, und mit echtem Verständnis hat es nichts zu tun. Die Validität der Prüfungen in Systemsoftware ist also äusserst schlecht. In diesem Sinn sind die Prüfungen völlig untypisch für die ETH, zumindest für unsere Abteilung. In den meisten Fächern geht ohne Verständnis nichts, dafür sind vergleichsweise langweilige Fakten weniger gefragt. Ausnahmen gibt es überall, doch der Befund ist klar.

In der letzten Prüfung hat Prof. Mössenböck 12mal eine Note unter 4.0 verliehen (das entspricht einer 'Versagerquote' von 18%), darunter gleich zweimal eine 2.5. In zumindest einem Fall ist diese Note für uns fragwürdig, da wir die normalen Leistungen des Kandidaten recht gut kennen.

Wir wünschen uns daher, dass Professor Mössenböck etwas mehr Nach- und Rücksicht bei der Vergabe der Schlussdiplomsnoten walten lassen würde. Denn eine solche Note im Schlussdiplom ist nicht nur sehr hässlich für den Kandidaten, sie repräsentiert in den seltesten Fällen sein wirkliches Leistungsvermögen.

Eine Bitte an Professor Mössenböck: Denken Sie etwas mehr an die Tragweite einer solchen Note, bevor Sie sie verleihen. Es mag Fälle geben, in denen eine 2.5 wirklich gerechtfertigt ist, aber bei Ihnen taucht sie zu häufig auf (und ist dann aus den genannten Gründen nicht akzeptabel). Ausserdem sind Sie der einzige, der nach so einer langen Studienzeit überhaupt noch solche Noten vergibt. Wir bitten sie daher eindringlich, Ihren Massstab zu revidieren.

Der VIS-Vorstand

Ranglist

ACM Scholastic Programming Contest
ETH Regional Contest

November 6, 1993

1.	Ceres - Quäler	ABFG	9.13
2.	Sogrjf	ABFG	11.12
3.	Noname	ABF	
4.	The Four	AG	
5.	Die Glorreichen vier	B	2.49
6.	Lyceanos Selection	G	3.41
7.	???	G	4.14
8.	C. Ray	A	4.28
9.	Maniacs		
9.	E 34		
9.	RATM		
9.	Postcondinion Q		
9.	Malaka		

Programmieren, bis der Compiler dampft...

Am 6.11.93 traf sich eine mehr oder minder muntere Schar von Informatikstudierenden frühmorgens um acht Uhr im Hörsaal HG F1 zu der vom VIS organisierten ETH-internen Ausscheidung für den weltweiten ACM-Programmierwettbewerb.
Frank rekapitulierte noch einmal die Regeln und verkündete, dass wir die beiden besten Viererteams an die westeuropäische Ausscheidung nach Swansea schicken würden. Dank der Grosszügigkeit der Sponsoren können die Kosten für Reise und Unterkunft vom VIS übernommen werden.
Nach dieser Orientierung ging es in den mit Plakaten und Bannern der Sponsoren versehenen Sun-Raum im G-Stock. In den kommenden fünf Stunden galt es nun, in Teams von drei Leuten – einer hielt sich als Reserve zur Verfügung – von den sieben Aufgaben so viele wie möglich richtig zu lösen. Dafür stand pro Team ein Computer zur Verfügung, auf dem in C oder PASCAL programmiert werden konnte. Die Abgabe der Lösungen erfolgte dank Netzwerk recht zügig.
Die Art und der Schwierigkeitsgrad der Aufgaben war recht verschieden: Die Überprüfung der Richtigkeit eines Kreuzworträtsels stellte (ausser der Jury...) niemandem allzugrosse Probleme. Auch das Haus des Nikolaus (gefragt waren alle Varianten, eine Figur mit einem Stift ohne absetzen zu zeichnen) wurde häufig gelöst. Die existentielle Frage "Dead or not Dead" des Schachspiels beschäftigte ebenfalls einige: Hier musste bestimmt werden, ob der König schachmatt steht oder nicht.
Die anderen Probleme, so die Textformatierung, das Rechnen mit beliebig langen Zahlen oder der Textvergleich erfreuten sich keinerlei Beliebtheit. Das lag aber vermutlich eher an der fehlenden Zeit denn an der Schwere der Probleme.
Nach fünf Stunden Denken und Programmieren machte sich in den Gesichtern der Teilnehmer sichtliche Erleichterung breit. Ob das am Apéro oder an der frischen Luft lag (im HG G26 herrschte gegen Ende der Veranstaltung tropisches Klima...), lässt sich nicht sagen. Trotzdem wurde noch über die Probleme eifrig diskutiert.
Um halb drei begann dann die mit Spannung erwartete Rangverkündigung. Nach den einleitenden Worten der VIS-Präsidentin Grete, die den Teilnehmern für ihr Kommen und dem OK und allen Helfern für ihren Einsatz dankte, ergriff Frank das Wort und nahm die Rangierung vor. Sieger wurde das Team "Ceres-Quäler" mit Patrick Aschwanden, Manuel Bleichenbacher, Erich Oswald und Renato Pajarola, gefolgt von den "Sogrjf" mit Rory Chisholm, Jörg Derungs, Daniel Friederich und Roland Ulber. Diese Teams werden also die ETH in Swansea vertreten; wir wünschen viel Glück!
Alles in allem sehen wir vom OK

diese Veranstaltung als gelungen an und hoffen, dass dies der Auftakt zu einer Tradition werden wird (die letzte ACM-Ausscheidung an der ETH fand 1989 statt).

(hd)

P.S.: Ausführliche Berichte über diese Veranstaltung und die Tage in Swansea werden in den nächsten Visionen zu finden sein.

[Ein kleiner Wermutstropfen: Keine einzige Frau nahm am Wettbewerb teil. Schade – ist Programmieren eine Domäne von Männern? Red.]

Was ist der Unterschied zwischen Jurassic Park und Microsoft?

Bei dem einen versucht ein reicher Irrer mit Dingen Geld zu machen, die eigentlich seit ewiger Zeit ausgestorben sein sollten und das andere ist ein Film von Steven Spielberg.

Aus de.talk.jokes

VIS-T-Shirt Reference Guide

Seit Anfang Semester ist die neue Auflage der VIS-T-Shirts draussen. Damit keine Missverständnisse erwachsen, geben wir hier exklusiv das offiziell gültige Bedienungshandbuch dazu durch.

1. Pull garment on over head, placing arms through appropriate openings
2. Finish with label at back of collar with design facing out
3. Wear shirt to pre-determined occasion (Important note - remove all tags or labels, before wearing in public)
4. After shirt is sufficiently soiled place in washing machine (Note - for best results remove shirt)
5. Leave the shirt just the way you removed it - inside out, wash warm water/cool rinse (Note - VIS shirts are NOT underwear - don't let your mother throw it in hot water and keep it away from the bleach)
6. Dry on low heat, air or line dry
7. Return garment to right-side out and repeat step 1

Für Fragen steht im IFW B29 eine Hot-Line zur Verfügung. Mittags von 12.15 bis 13.00 ist sicher jemand da.
Tel.: 2 72 12

Ausschnitt aus de.talk.jokes

Programmschutz durch das neue Urheberrecht

"Das Recht hat der Technik die Schranken zu setzen und nicht umgekehrt." (*sic!*)
Prof. Dr. Karl Oftinger, Rechtsgelehrter, Festschrift zum Zentenarium des Schweizerischen Juristenvereins, Zürich 1961

Am 1. Juli 1993 trat in der Schweiz das revidierte Urheberrechtsgesetz (URG) in Kraft, das neu auch Computerprogramme explizit schützt. Im folgenden werden die wesentlichen Fragen aus Sicht der Informatik diskutiert.[2]

Warum ins Urheberrecht ?
Verglichen mit dem Patentrecht ist das Urheberrecht besser geeignet für den Schutz von SW: Rasch (gilt sofort ab Werkfestlegung), unbürokratisch (keine Registrierung), diskret (keine Offenlegung), kostenlos und einfacher zu erlangen (keine "Erfindungshöhe" nötig).
Jedoch war lange umstritten, ob man wegen der speziellen Eigenschaften von SW nicht ein neues, spezielles Sonderschutzrecht für SW ausarbeiten solle. Aus Sicht der Informatik wäre dies wünschenswert gewesen (s. Kritik am Ende des Artikels), doch wegen der internationalen Gesetzeslage entschied sich der Gesetzgeber für die Integration ins URG, um keine Schutzbenachteiligung schweizerischer Programme im Ausland zu riskieren.

Schutzgegenstand
Neben dem fertigen Programmcode sind auch Entwürfe, Titel und Teile von Programmen schützbar; als Teile gelten auch Datenstrukturen und I/O-Gestaltung. Schutzgegenstand kann immer nur die *Form* (bei Programmen die *konkrete* Implementation und I/O-Gestaltung) sein; der *Inhalt* (die zugrundeliegenden Algorithmen und Ideen) muss hingegen schutz*frei* bleiben für die technische Entwicklung und die geistige Auseinandersetzung in Forschung und Ausbildung. (Dijkstra geht also materiell leer aus...) Der Schutz von Algorithmen und Ideen wurde in den Bereich des Geheimnisschutzes (Wettbewerbsrecht) verwiesen.
Konkret: Ein geschütztes Code-Stück wörtlich abzuschreiben, oder ein Piktogramm genau abzukupfern, ist verboten (ausser, dessen Urheber ist einverstanden); hingegen nur dessen Algorithmen bzw. Gestaltungsideen zu übernehmen (aber anders zu codieren), ist erlaubt.
Wo genau die Grenze zwischen Form und Inhalt gezogen werden soll (oft macht doch gerade die *Einheit* von Form und Inhalt das Meisterwerk aus), ist trotz jahrzehntelangen Kontroversen bis heute nicht genau geklärt.

2 Eine umfassende Abhandlung findet sich im "EDV&Recht"-Seminarbeitrag "Analyse von Gegenstand und Umfang des Programmschutzes" (40 S.) des Verfassers, in der Informatik-Bibliothek.

Schutzkriterien

Um urheberrechtlich geschützt zu sein, muss ein Programm(teil) im wesentlichen folgende zwei Kriterien erfüllen:

- Eigene geistige Schöpfung des Urhebers. Der Urheber kann weder für (aus anderen Werken oder dem Allgemeingut) übernommene Teile noch für aleatorische (durch Zufallsgeneratoren erzeugte) Komponenten Schutz beanspruchen.
- Individueller Charakter. Gerade bei Programmen gab es darüber jahrzehntelang fast so viele Meinungen wie Rechtsdogmatiker. (Um das Chaos perfekt zu machen, wurden in Rechtsprechung und Lehre die Begriffe «Individualität», «Originalität» und «eigenpersönliche Schöpfung» vermischt und/oder je unterschiedlich gedeutet.) Eine Definition umschreibt Individualität als Abweichen vom Vorbestehenden und vom zu Erwartenden; eine andere Definition (statistische Einmaligkeit) ortet Individualität dort, wo die Wahrscheinlichkeit *praktisch* Null ist, dass ein anderer Urheber *un*abhängig genau dasselbe Werk schafft. Individualität bedeutet *nicht*, dass das Werk die Persönlichkeit des Urhebers widerspiegeln muss; der individuelle Charakter ist ausschliesslich im Werk selbst zu suchen.

Wichtig ist, dass die Anforderungen an die Individualität abhängig vom vorhandenen *Gestaltungsspielraum* sind: Je grösser dieser ist, desto spezieller muss ein Programm(teil) sein, um Schutz zu erlangen. Wo hingegen gar kein Spielraum besteht, ist kein Schutz möglich, weil ja jeder Urheber dort zur selben Lösung gelangen muss. Der an sich (durch die Komplexität) grosse Gestaltungsspielraum bei Programmen wird in der Praxis eingeschränkt durch Effizienz-Anforderungen, technische Möglichkeiten, Normen/Richtlinien, und nicht zuletzt Erwartungen der Kunden.
Ansatzpunkte für Individualität ergeben sich z.B. bei der Benutzerschnittstelle aus folgenden Gestaltungsmöglichkeiten:

- I/O-Masken und -Layouts (grafisch oder textuell);
- Fehlerbehandlung und -meldungen, Hilfstexte;
- grafische Details (Piktogramme; Hervorhebungen, Hell/dunkel-Abstimmungen, Färbungen etc.);
- Editiermöglichkeiten (z.B. bei einem 3D-Editor: Verschiedenartige Perspektiven, Ausschnitte, Weglassungen, Zooms, etc.);
- akustische Unterlegung.

In der Rechtsprechung wird ein Werk immer nach der *Gesamtheit* aller seiner Elemente beurteilt. Dies ändert nichts daran, dass jene Teile, welche die Schutzanforderungen nicht erfüllen, frei übernommen werden dürfen, ist aber relevant für Raubkopierer (für deren Illegalität es bereits genügt, dass einzelne Elemente des Programmes geschützt sind).

Nicht relevant für den Programmschutz ist die Art der Festlegung (ob als Source- oder Object-Code; verwendete Programmiersprache; Datenträger) sowie der Wert (wirtschaftlich, praktisch, theoretisch), Entwicklungsaufwand (Zeit, Geld), Umfang, Zweck (Bestimmung) und Ästhetik des Programmes (letztere ist Geschmackssache und somit ungeeignet für die Rechtsprechung).

Schutzumfang

Die Rechte des Urhebers umfassen:

- *Persönlichkeitsrechte* (exklusiv an die natürliche Person des Urhebers gebunden): Recht auf Anerkennung als Urheber (d.h. auf *Vergabe* der Verwendungsrechte), Werkerstveröffentlichung (ob, wann, wie und unter welcher Urheberbezeichnung) und Werkintegrität (Schutz vor Entstellung).
- *Verwendungsrechte* (können an Arbeit-/Auftraggeber, Erben und z.T. Lizenznehmer des Urhebers übergehen): Werk-Verwendung (ob, wann, wie), Werk-Änderung/-Weiterentwicklung/ -Übersetzung, Werk-Aufnahme in ein Sammelwerk (Programmpaket o.ä.) sowie das Herstellen, Anbieten, Vermieten, Veräussern, Senden oder anderweitige Verbreiten/Wahrnehmbarmachen von Werkexemplaren (via Netzwerk, Datenträger, Listings etc.).

Die Rechte gelten von der Werkfestlegung bis 50 Jahre nach dem Tod des Urhebers (Persönlichkeitsrechte logischerweise nur bis zum Tod). Bei Kollektivwerken stehen die Rechte allen Miturhebern gemeinschaftlich zu. Die Verwendungsrechte an *innerhalb eines Anstellungsverhältnisses* geschaffenen Programmen gehen automatisch an den Arbeitgeber über.

Im neuen URG gelten Verstösse nicht mehr als Kavaliersdelikte: Es winken bis zu 1 Jahr Gefängnis und/oder Fr. 20000.– Busse; gewerbsmässige Vergehen werden vom Staatsanwalt verfolgt und mit bis zu 3 Jahren und/oder Fr. 100000.– geahndet.

Werke zweiter Hand

Entwickelt z.B. ein Programmierer B ein Programm PA des Programmierers A durch Hinzufügung neuer Optionen (oder Anpassung an eine andere Umgebung) weiter zu einem Programm PB, so muss B von A die Erlaubnis zur Werkbearbeitung einholen, und einen Teil der aus PBs Verwertung erzielten Einkünfte an A abliefern. Am resultierenden Programm PB haben dann A und B gemeinsam Urheberrechte.

Gemäss dem oben Gesagten zu Form vs. Inhalt (Inhalt ist *frei*) liegt ein Werk zweiter Hand *nicht* schon dann vor, wenn PA in PB irgendwie als Vorlage erkennbar ist (d.h. nur Ideen/Algorithmen von PA übernommen wurden); vielmehr muss PAs *Individualität* in PB zutage treten, damit A an den Rechten an PB beteiligt wird.

Werk-Parodieren ist urheberrechtlich frei, was vielleicht für gewisse Benutzeroberflächen relevant ist...

Rechte der Benutzer

Der Benutzer hat folgende Rechte am legal erworbenen Programmexemplar:

- Bestimmungsgemässe Verwendung (wie vom Lizenzgeber vorgesehen).
- Anlegen einer Sicherungskopie für die Dauer der Lizenz. (Dies kann in nach dem 1. Juli 1993 geschlossenen Verträgen *nicht* untersagt werden.)
- Weiterveräusserung, wenn anschliessend die eigenen Exemplare gelöscht werden.
- Reverse Engineering: Verwendungsberechtigte dürfen sich die *Schnittstelleninformationen* eines Programms durch Analyse des Programmcodes beschaffen (lassen), soweit sie diese für die Herstellung der Interoperabilität mit anderen Programmen benötigen, und soweit diese nicht anders (z.B. direkt vom SW-Hersteller) erhältlich sind. Die so gewonnenen Informationen dürfen jedoch nicht zum Nachmachen des analysierten Programms missbraucht werden.

Konsequenzen für die Praxis

Für die SW-Hersteller hat das neue URG zwei Aspekte: Einerseits werden ihre eigenen Programme nun explizit geschützt, andererseits müssen sie nun vorsichtiger sein, wenn sie Teile aus fremden Programmen übernehmen. Die recht diffusen Schutzbestimmungen (Individualität, Form/Inhalt) eröffnen der Rechtsprechung einige Spielräume bei der Beurteilung der Schutzfähigkeit.

Um die Schutzmöglichkeiten voll auszuschöpfen, wird konkret empfohlen:

- Klare Verträge mit Anwendern, Arbeitgebern, externen Testern etc.: Unklarheiten und Lücken des URG ausbügeln mittels *vertraglicher* Regelungen (z.B. betreffend Programm-Gebrauch auf portablem Zweitgerät und im Netzwerk, Rechtsübergang bei Gelegenheits-Programmen im Angestelltenverhältnis,
  Geheimhaltungsvorschriften,
  Kontrollrechte).
- Copyright-Vermerk und Markennennung am Bildschirm sowie in Source- und Object-Code und Anwenderdokumentation.
- Umfassende (interne) Dokumentation auf allen SW-Entwicklungsstufen zwecks Beweisführung.
- Einbau versteckter Erkennungsmerkmale in den Objectcode, als allf. Beweismittel vor Gericht.

Kritik

Meines Erachtens ist das Urheberrecht *grundsätzlich ungeeignet* für den Schutz von Computerprogrammen. Das Urheberrecht wurde Ende des 19. Jahrhunderts für Werke der Literatur und Kunst konzipiert entsprechend sind die Schutzbestimmungen gewählt. Dem Dichter/Künstler geht es darum, sich auszudrücken, dem Programmierer hingegen um ein technisch/ökonomisch optimales Produkt (Stichwort "Software *Engineering*"). Für Programme sind deshalb *andere Schutzkriterien* nötig:

- Das Schutzkriterium sollte bei SW *nicht* Individualität, sondern Normierung und Funktionalität sein: Sonst wird nämlich chaotischer, dirty Programmierstil (eben besonders individualistisch, barock) *eher* geschützt als sauberer, normgerechter (und somit *un*individueller) Programmierstil! Analoges gilt beim "look and feel" (die Benutzer wollen *Einheitlichkeit* statt verwirrende Vielfalt) und bei der Dokumentation (in Handbüchern ist episch-schnörkelige Dichtkunst kaum gefragt, sondern schlichte Funktionalität). Individualität im Sinne von Abweichen vom Vorbestehenden und zu Erwartenden ist gerade bei fortlaufenden Versionen des selben Programmes eben genau *nicht* sinnvoll.
- *Marktrelevante* SW-Eigenschaften (Leistungsfähigkeit, Benutzerfreundlichkeit, Ästhetik, Kompatibilität) sind *keine* Schutzkriterien des Urheberrechts.

Auch bei anderen Schutzbestimmungen sprengen die *speziellen Eigenschaften* der SW den Rahmen des Urheberrechts:

- Utilitaristischer Charakter und leichte originalgetreue Kopierbarkeit der SW machen künstlich aufgesetzte Ausnahmebestimmungen bei privatem Eigengebrauch und Vermietung nötig (bei allen anderen Werkkategorien frei bzw. erlaubt, bei Programmen lizenzabhängig bzw. verboten).
- Weil Programme nie fertig sind (Fehlerkorrekturen, Weiterentwicklungen, Anpassung an neue Umgebungen), und meist im Auftrag erstellt werden, sind die im Urheberrecht verankerten exklusiven Persönlichkeitsrechte im Wege: Sie gehen nicht an den Auftrag-/Arbeitgeber über, weshalb der Urheber z.B. für die Erstveröffentlichung und für jede Änderung um Erlaubnis gefragt werden muss.
- Das Werkschaffen in Teams und die schwierige Trennbarkeit der Einzel-Beiträge können v.a. bei Personalfluktuationen bald einmal Unklarheiten bei der gemeinschaftlichen Rechtsausübung verursachen.
- Die urheberrechtliche Schutzdauer (zwischen gut 50 und ca. 120 Jahren) ist viel zu lang für die schnellebige SW.

Fazit

Bedauerlich, dass sich der Gesetzgeber nach jahrzehntelangen Programmschutz-Kontroversen aus rechtssystematischen Gründen international für die aus Sicht der Informatik unbefriedigende Integration des Programmschutzes ins Urheberrecht entschieden hat.

Christoph Reuss, IIIC/7

Der Gesetzestext zum neuen URG ist bei der EDMZ in Bern erhältlich (Nr. 33.314d, Fr.

FIGUGEGL

Fondue isch guet und git e gueti Lune

Datum: Donnerstag, 2. Dez.

Ort: StuZ - Saal
(Leonardstrasse 19)

Zeit: 20:00 Uhr

Unser Angebot:
Fondue soviel ihr wollt
Wein
Snacks und viel zu trinken

FIGUGEGL

Fondue isch guet und git e gueti Lune

Dieses Kleingedruckte traegt nichts zum Inhalt bei :-) Ja, das hab ich gemerkt!.. pel

Statistiken zu den Prüfungen der Herbstsession

Manche von Euch werden es bei diesem Titel schon erahnen: Im Spätsommer/Frühherbst dieses Jahres waren wieder einmal Prüfungen. Die Ergebnisse kommen immer etwas später. Der Artikel in den Visionen dazu jetzt. Jetzt ist die Zeit reif, jetzt sollt Ihr (beinahe) alles über den Ausgang dieser alljährlich wiederkehrenden Schlacht erfahren:

Vordiplomen). Erfreulicherweise haben sich die Durchfallquoten im Grundstudium nicht verschlechtert. Die Durchfallquoten im ersten Vordiplom stagnieren seit Jahren, im zweiten Vordiplom sind sie allerdings auf 35 % gesunken (das ist das schönste Ergebnis, noch dazu, wo - abgesehen vom diesjährigen Frühling - erstmals seit Frühjahr 1989 die Durchfallquote im zweiten Vordiplom wieder kleiner als im ersten ist. Darauf heben wir ein Glas - hoffentlich können wir das noch öfter tun). Im ersten Teil des Schlussdiploms hat sich für die Herbstsessionen allerdings ein neues Durchfallsmaximum seit Herbst 1989 ergeben.

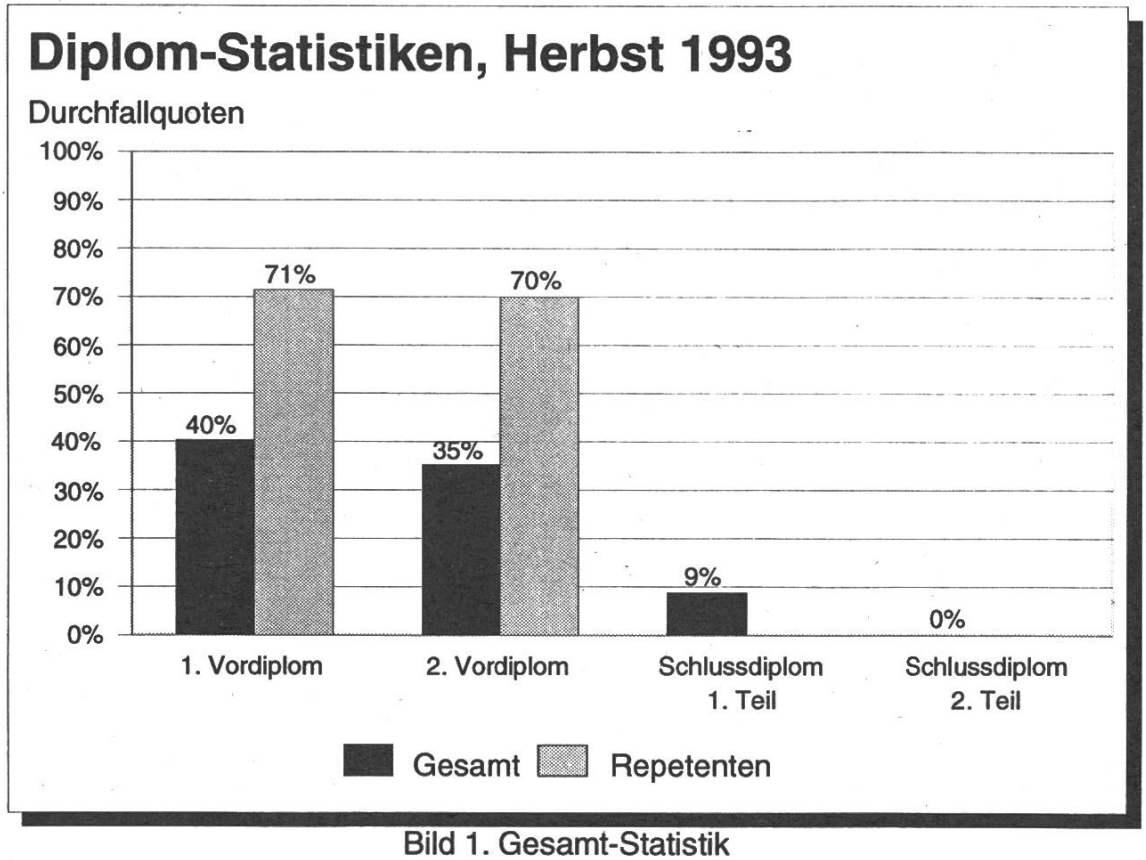


Bild 1. Gesamt-Statistik

Bild 1 soll Euch auf einen Blick die Durchfallquoten der diesjährigen Herbstsession zeigen: Wie ihr seht: Das war dieses Jahr alles nicht so schlimm (in den

Darüber will ich mich jetzt nicht auslassen, da gibt es andere, die das besser können. (Aber dass so etwas deprimiert, das dürft Ihr mir glauben.) Ich möchte

nun hingegen lieber zur Besprechung der Einzeldisziplinen übergehen:

1. Vordiplom

Anzahl Kandidaten	119
davon Repetenten	14
Bestanden	71 = 60 %
davon Repetenten	4 = 29 %
Durchschnitt	4.16
Varianz	0.56

Fächerstatistik	Schnitt	Varianz
Analysis I+II	4.45	1.14
Algebra I+II	4.33	0.98
Elektrotechnik I+II	4.21	0.75
Informatik I+II	3.89	1.14
W'keit & Statistik	3.88	1.74

Eigentlich war an dieser Stelle ein Vergleich mit früheren Herbstsessionen geplant. Leider habe ich keine verlässlichen Daten der früheren Jahre. Es hat wenig Sinn, die letzte Frühjahrssession diesen Ergebnissen gegenüberzustellen, da im Frühjahr ganz andere Verhältnisse bei den Prüfungen herrschen (z. B. weniger Leute, mehr Repetenten).

Die Repetentendurchfallquote ist sehr hoch. Das war sie schon immer. Aufgrund der kleinen Repetentenzahlen ist sie nicht sehr aussagekräftig (sie war in den Herbstsessionen immer grossen Schwankungen unterworfen). Ich werde mich im Frühling etwas ausführlicher mit ihnen befassen.

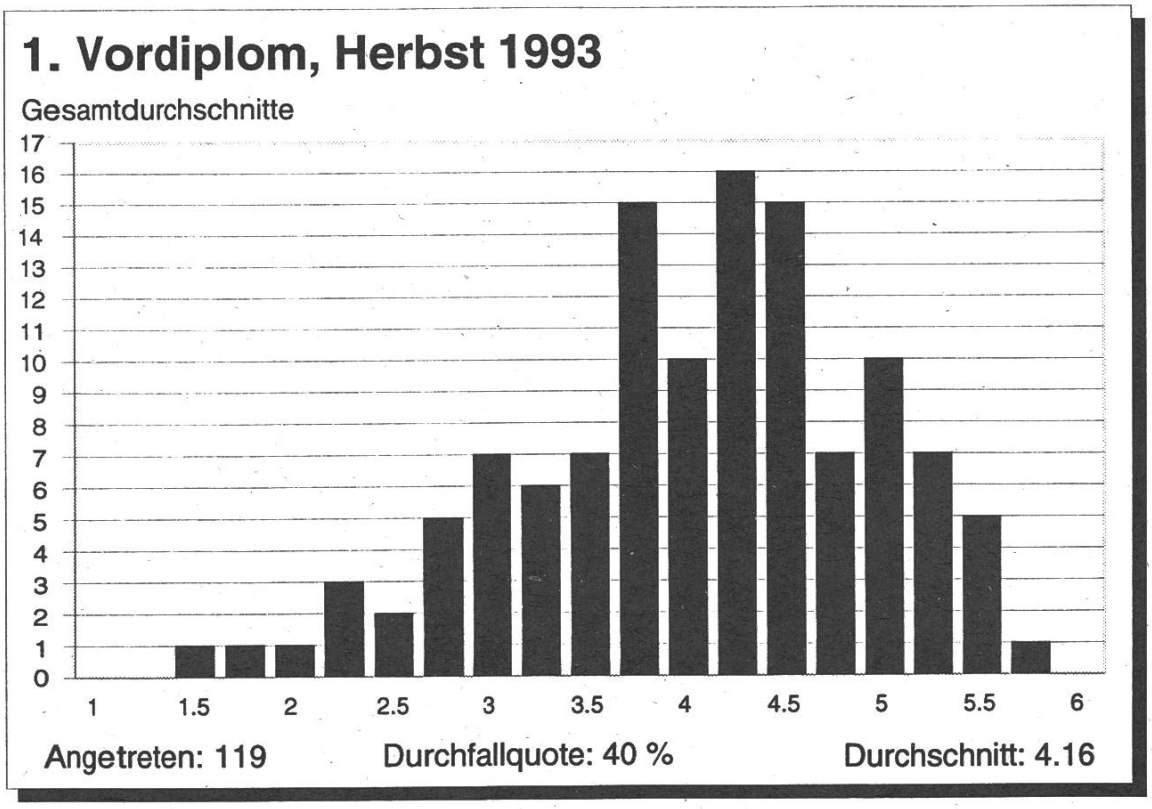


Bild 2. Histogramm, 1.Vordiplom

Da das erste Vordiplom in den Herbstsessionen bisher meistens eine Durchfallquote um die 40 % aufgewiesen hat, ist dieses Ergebnis als normal zu bezeichnen. Vielleicht liessen sich aber gerade hier die Quoten noch senken, indem eine qualitativ bessere Vorlesung in Wahrscheinlichkeitsrechnung & Statistik[3] angeboten werden würde.

2. Vordiplom

Anzahl Kandidaten	105
davon Repetenten	20
Bestanden	66 = 65 %
davon Repetenten	6 = 30 %
Durchschnitt	4.16
Varianz	0.79

Fächerstatistik	Schnitt	Varianz
Theor. Informatik I+II	4.48	1.00
Informatik III+IV	4.39	1.00
Elektrotechnik III+IV	4.28	0.52
Wissens. Rechnen I+II	3.86	0.66
Physik	3.78	1.21

Die glorreiche Nachricht über das zweite Vordiplom habe ich schon verkündet. Generell sollte die Durchfallquote bei dieser Prüfungsreihe kleiner sein als im ersten Vordiplom (das ist ein Gebot der Fairness). Dass dieses Gebot seit zwei Sessionen wieder einmal eingehalten wird, ist wirklich positiv hervorzuheben.

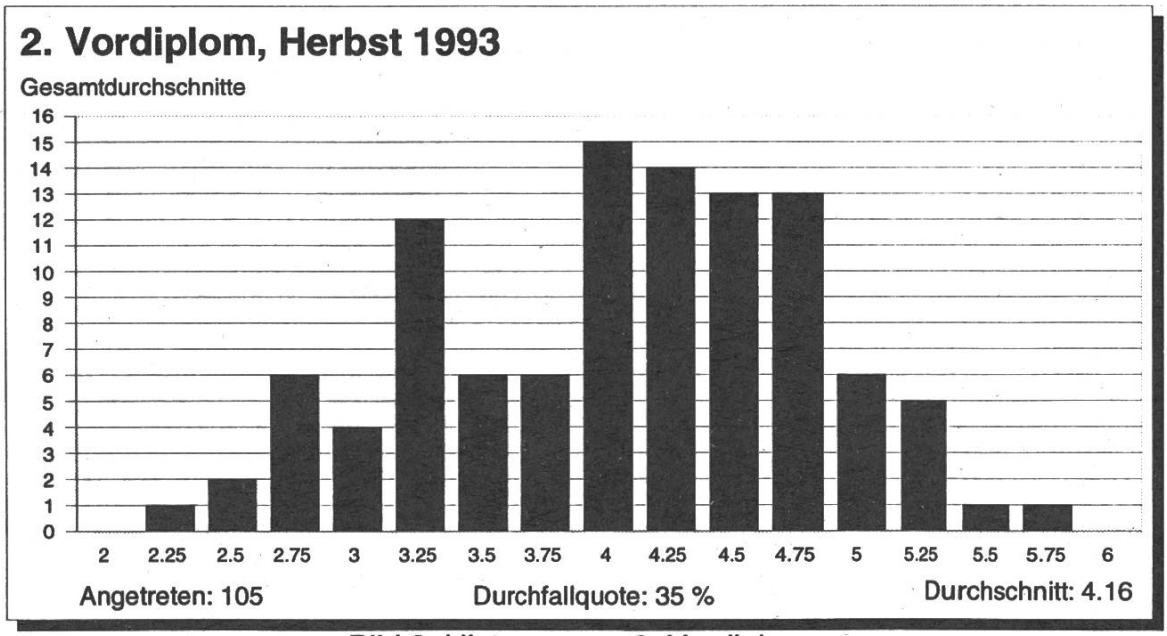


Bild 3. Histogramm, 2. Vordiplom

3 Grundlage für diese Vermutung ist die letzte Vorlesungsumfrage

Unerfreulich ist weiterhin die Physik, "Wissenschaftliches Rechnen" befindet sich leider auch im negativen Bereich. Die Informatik hat sich durch ein positives Zeichen hervorgetan: War sie in vergangenen Sessionen eher am Schluss zu finden, so hat sie in dieser Session (mit einem zweiten Platz hinter der Theoretischen Informatik) das Ihre zur guten Durchfallquote beigetragen.

Für die Repetentendurchfallquote gilt, was bei der Besprechung des ersten Vordiploms gesagt wurde: Im Frühling erfahrt ihr mehr darüber. Nehmt sie zur Kenntnis aber nicht so ernst. Mit 20 Leuten lässt sich wirklich keine aussagekräftige Statistik machen.

Schlussdiplom, 1. Teil

Anzahl Kandidaten	68
davon Repetenten	k. A.
Bestanden	62 = 91 %
davon Repetenten	k. A.
Durchschnitt	4.68
Varianz	k. A.

Fächerstatistik	Schnitt	Varianz
Rekurs. & Kompl.	4.48	1.00
Digitalt. & Rechnerstr.	4.39	1.00
Informationssysteme	4.28	0.52
System-Software	3.86	0.66

Fortsetzung auf S. 30

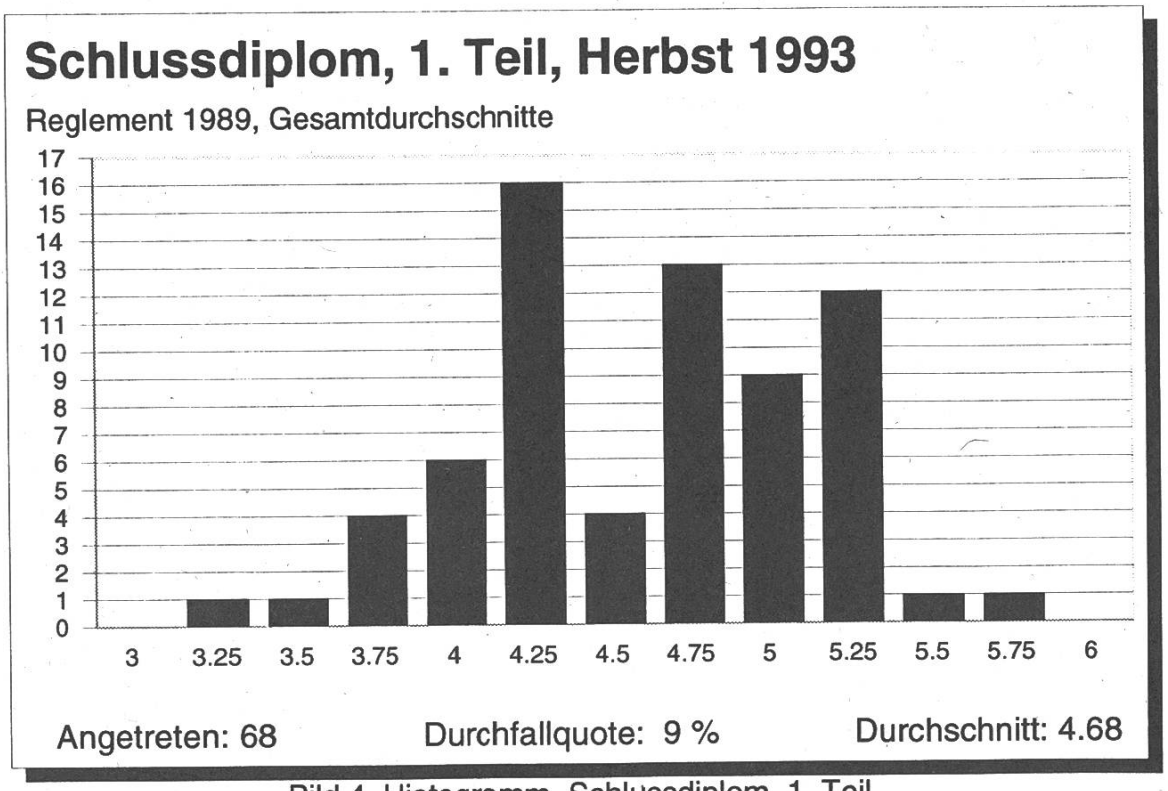


Bild 4. Histogramm, Schlussdiplom, 1. Teil

Who is who

Professor Yuri Breitbart is a Gastprofessor in the Institute of Information Systems, Department of Computer Science, ETH-Zentrum, Zurich, Switzerland. He is on sabbatical leave from the Department of Computer Science University of Kentucky, Lexington, Kentucky USA, where he permanently resides.

Professor Breitbart has received his Doctor of Science in Computer Science in 1973 from Computer Science Department of Israel Technological Institute (TECHNION), Haifa, Israel. His main research interest is in the area of distributed databases. He published extensively in the area of distributed heterogeneous databases.

At ETH Professor Breitbart works in the area of transaction management models and distributed file structures. His work is conducted in close cooperation with Professors Schek and Weikum from the Institute of Information Systems.
Professor Breitbart serves as a reviewer for various computer science journals, including IEEE Transaction on Computers; Communication of ACM;
Journal of ACM; IEEE Transactions on Knowledge Engineering; Transactions of ACM on Database Systems; and Computing Surveys.

He is a member of Editorial Board for Distributed and Parallel Databases International Journal and a receipient of several National Science Foundation research grants.

During his sabbatical at ETH Professor Breitbart has taught a Multidabase System course (summer semester 1993) and he is currently teaching a Concurrency Control Theory course.
(Series *Who is who* to be continued – send us your texts! Ed.)

Nun kommen wir zum schwarzen Schaf der diesjährigen Herbstprüfungen. Nicht nur, dass die Durchfallquote recht hoch ist, nein, auch dieses Jahr ist mindestens einE KandidatIn zum zweiten Mal durchgefallen. So etwas tut nun wirklich nicht not. Das Schlussdiplom darf nicht zum Sieben verwendet werden! Da wäre es doch wirklich fairer, wenn man im ersten Vordiplom die Ansprüche etwas hebt.
Erstaunlicherweise hat gerade das beliebteste Fach[4] den schlechtesten, das unbeliebteste den besten Schnitt. Bemerkenswert: Im Frühling 1993 hatte kein einziges Fach einen Schnitt unter 4.4, in der jetzigen Session lag einzig und allein Rekursivität & Komplexität über dieser Marke.
Da im Frühling der Studienplan wechselt, möchte ich zu den Kernfächern noch eine zusätzliche Graphik zeigen:

Bild 5 gibt einen Ausblick auf das Reglement 1993. Wenn in der jetzigen Prüfungssession dieses Reglement schon in Kraft gewesen wäre, hätte sich eine Durchfallquote von 18 % ergeben. Hoffentlich wird das im Frühling besser - im Grunde kann man sich dann ja auf zwei statt auf vier Fächer konzentrieren.

Schlussdiplom, 2. Teil

Alle Kandidaten (69) haben den zweiten Teil der Schlussdiplomprüfung bestanden. Das ist erfreulich (aber es wäre tragisch, wenn dem nicht so wäre). Sonst gibt es da nichts zu sagen (wir haben auch keine anderen Daten bekommen).

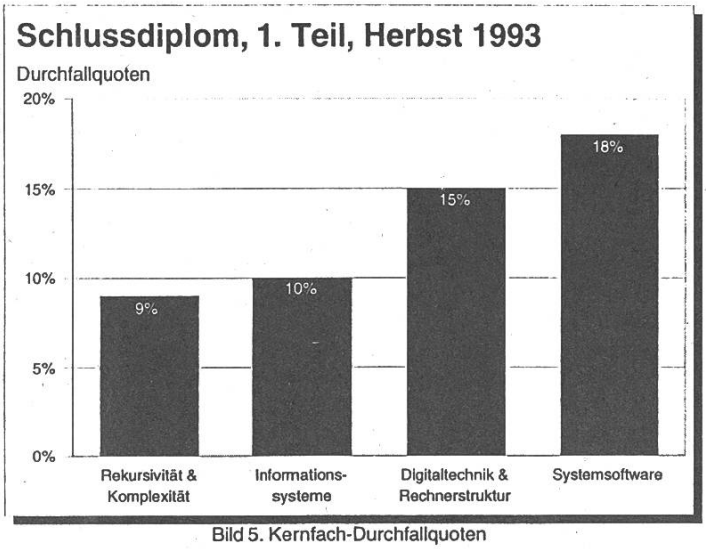
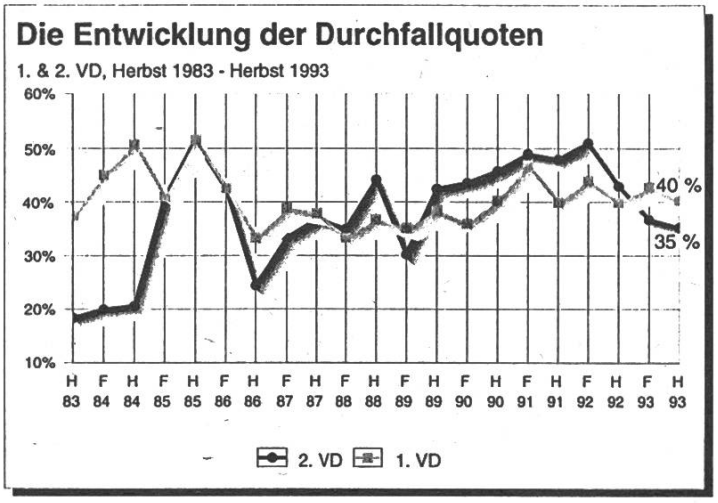
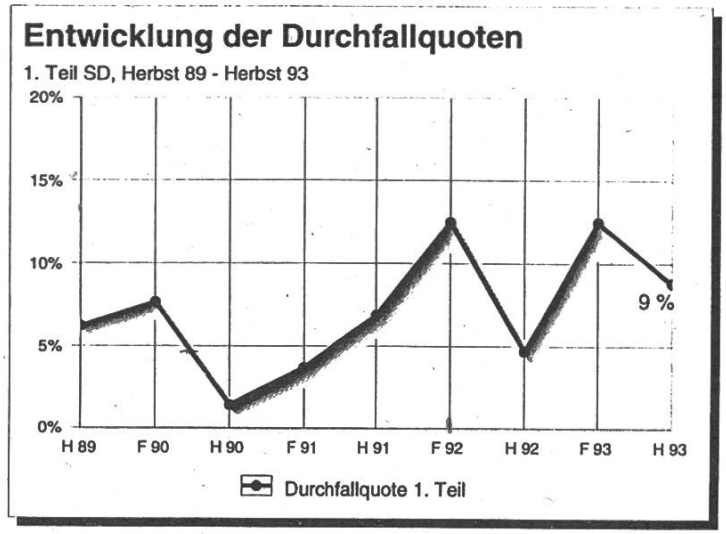


Bild 5. Kernfach-Durchfallquoten

4 Grundlage ist die letzte Vorlesungsumfrage

Zum Schluss möchte ich Euch einen relativ unkommentierten historischen Rückblick auf die Durchfallquoten geben. Dieser soll es erleichtern, die jetzt vorgestellten Resultate im richtigen Kontext zu sehen. Ausserdem bin ich Euch zumindest das schuldig, da ich bei den speziellen Ergebnissen keine Vergleichsmöglichkeiten geben konnte (wenigstens die Durchfallquoten konnte ich ein paar Jahre zurückverfolgen). Im Schlussdiplom (1. Teil) geht der Rückblick nur bis ins Jahr 1989 zurück. Vorher versickern die Daten in den Visionen.





Die Vordiplome sind also recht gut ausgefallen. Das Schlussdiplom, erster Teil, macht mir ein wenig Sorgen. Da im Frühling 1994 die erste Prüfung nach dem neuen Reglement fällig ist, ist hier auch die grösste Unsicherheit: Wie werden die Schlussdiplom-Prüfungen im Frühling ausfallen? Hoffen wir das Beste - was können wir sonst schon tun?

lj

Perspektiven mit ABB

Informatik-Praktika

ABB Schweiz mit mehr als 35 selbständigen Tochtergesellschaften entwickelt, erzeugt, verkauft und wartet Systeme und Produkte eines breiten Sortimentes zur Bereitstellung und Anwendung elektrischer Energie. Angehenden Informatik-Ingenieuren und -Ingenieurinnen steht damit auch eine breite Palette von Praktikumsmöglichkeiten offen:

System-Software
- Graphische Programmierung
- Compilerbau
- Betriebssysteme

Verteilte Systeme
- Kommunikation
- Prozesssteuerung
- Netzleitsysteme

Datenbanken
- Engineering-Datenbanken
- Nichtstandard-Datenbanken

Wissensbasierte Systeme
- Expertensysteme für Konfiguration und Diagnose

Auf diesen Gebieten arbeiten wir in internationalen Teams an interessanten Projekten. Im Rahmen eines Praktikums haben Sie Gelegenheit, dabei mitzuwirken, persönliche Erfahrungen zu sammeln und Einblick in die Berufswelt zu gewinnen.

Weitere Auskunft und Unterlagen bei

Ruth Maurer Telefon 056/75 20 56 oder
Dieter Spickenreuther Telefon 056/75 63 31

ABB Management AG
Abteilung PMZ
Postfach
5401 Baden

ABB

Wie werde ich am einfachsten Guru?

oder: vi für Anfänger

Früher (es ist noch kein Jahr her) war ich jeweils einigermassen ratlos, wenn ich ein File editieren sollte und im Computerraum wieder mal nur ein Terminal frei war. Ich sah bewundernd Kommilitonen zu, die in solchen Situationen Zauberwörter wie *vi* oder *emacs* murmelten und mit obskuren Kommandi und zusammenhangslos über die Tastatur verstreuten Tastendrucken dem File geheimnisvoll die erwünschten Änderungen einhauchten...

Unterdessen habe ich mich bei meinem Praktikum wohl oder übel mit dem Editor vi herumschlagen müssen. Zwar konnte ich zu Beginn auch nur mit einer Quick Reference (A4-Blatt beidseitig klein bedruckt) in Griffnähe arbeiten, aber bald habe ich gemerkt, dass ein gutes Dutzend Befehle völlig ausreicht! Sie sind im folgenden Text fett gedruckt. Unterdessen weiss ich noch ein paar Kommandi mehr, vielleicht ein Viertel dessen, was auf der Quick Reference stand, also immer noch ein nur ein Bruchteil der vollen Grösse von vi. Aber als ich wieder einmal ein File mit vi las weil mir more zu unkomfortabel ist, und der Kommilitone daneben Augen und Mund aufgerissen hat ('Was, du kannst vi !?'), beschloss ich, mein bescheidenes Wissen niemandem vorzuenthalten. Man wird nämlich offenbar nirgends einfacher Guru als in vi!
Für den Anfang reichen die fettgedruckten Befehle völlig aus. Wenn du es komfortabler magst, werden dir ein paar der andern auch nützlich sein. Für besonders Angefressene: Mehr als hier steht, weiss ich auch nicht; aber in der Infobibliothek habe ich ein ca. 1,3 cm dickes Buch über vi gesehen...

Beginnen wir beim allerwichtigsten: Grundsätzlich gibt es in vi zwei Modi, nämlich den *Insert*-Mode – in ihm wird jedes eingetippte Zeichen auf den Bildschirm geschrieben – und den *Command*-Mode, welcher getippte Zeichen als Befehle interpretiert. Text schreiben kannst du nur im Insert-Mode, alle andern Operationen werden im Command-Mode ausgeführt; aber keine Panik, das beschreibe ich schon noch genauer.

Cursorbewegungen:
Zu Beginn befindet man sich im Command-Mode. In diesem muss man sein, um den Cursor zu bewegen. **Ctrl-b und Ctrl-f scrollen seitenweise rückwärts** (b wie backward) **bzw. vorwärts** (f wie forward). ***:n* gefolgt von [CR] springt zur Zeile *n***; ein sehr vielseitig anwendbarer Befehl, mit dem man allerdings nicht viel anfangen kann, wenn man gar nicht weiss, wo man sich befindet. **Ctrl-g** schafft da Abhilfe: Es zeigt auf der untersten Zeile **allerlei Informationen**, unter anderem die **aktuelle Zeilennummer und die Anzahl Zeilen** des ganzen Texts. Wenn man also ans Textende springen möchte, aber den Befehl G (springt auf die letzte Zeile) vergessen hat, hilft Ctrl-g, danach weiss man, welche Nummer die letzte Zeile hat und kann den obigen *:n* Befehl anwenden. Eigentlich sind das schon alle Cursorpositionierbefehle, die du unbedingt kennen musst... ausser in dem unglaublichen Fall, der jeden normalen Informatiker aus der Bahn wirft: den Cursortasten sind nur seltsame Zei-

chenfolgen zu entlocken, ohne dass sich jedoch der Cursor in irgendeine der gewünschten Richtungen bewegen würde. Wer jetzt noch weiter weiss, gilt bei vielen schon als vi-Guru. Dabei ist es ganz einfach: die **vier nebeneinanderliegenden Tasten h, j, k und l übernehmen die Funktion der Cursortasten.** Minimalisten wie ich merken sich nicht einmal, welche was tut, das merkt man ja dann schnell, falls man sie tatsächlich einmal benötigen sollte... Im weiteren springt 0 (Null) an den Zeilenanfang, $ ans Zeilenende (überhaupt hat $ immer etwas mit Ende zu tun, aber das kommt noch). C-Programmierer wissen den Befehl % zu schätzen: steht man nämlich auf einer Klammer (egal welcher Art), springt % zu ihrem Gegenstück (mit nochmaligem % ist man natürlich wieder am Ursprungsort); hilfreich bei der leidigen Frage: Wo fehlt die Klammer?... Fast schon Luxus, aber eben doch praktisch sind die Kommandi e (end) und b (begin) mit ihren Verwandten E und B, welche den Cursor wortweise vorwärts bzw. rückwärts bewegen. Bei e und b gilt dabei als Wortende ein Leer- oder Satzzeichen. Bei E und B dagegen zählt nur das Leerzeichen als Wortende, die 'Wörter' sind also oft länger und somit kommt man schneller in die gewünschte Richtung.

Mode-Wechsel, Text einfügen und überschreiben:
Eine der am häufigsten gebrauchten Tasten in vi ist wohl die **ESC-Taste**; sie **wechselt vom Insert-Mode zurück in den Command-Mode.** Wenn man sich schon im Command-Mode befindet und ESC tippt, passiert ausser einem Piepston weiter nichts; **im Zweifelsfall also mal [ESC] drücken**, danach bist du sicher im Command-Mode.
Um umgekehrt vom Command-Mode in den Insert-Mode zu gelangen, gibt es dagegen diverse Möglichkeiten. Die wohl einfachste ist i (insert): der Cursor bleibt auf seiner Position und **es wird auf Einfügen, also in den Insert-Mode umgeschaltet.** Ebenfalls oft benötigt ist **o** (open), welches **eine freie Zeile unterhalb der Cursorzeile eröffnet und dort mit dem Einfügen beginnt.** O (grosses o) dagegen eröffnet oberhalb des Cursors eine freie Zeile, ein nicht lebensnotwendiges, aber praktisches Kommando, ebenso wie A (append), welches am Ende der Zeile einfügt. R schaltet ebenfalls in den Insert-Mode, allerdings wird überschrieben anstatt bloss eingefügt, bis man den Insert-Mode wieder mit ESC verlässt. Recht nützlich ist cw (change word); es löscht die Zeichen von der Cursorposition bis zum Wortende (wird mit einem $ markiert) und ersetzt sie durch alles, was nach cw getippt wird (ebenfalls bis [ESC] folgt). Mit **r** wird **das Zeichen unter dem Cursor** durch das nächste eingetippte Zeichen **ersetzt**; ra ersetzt also das Zeichen an der Cursorposition durch ein 'a'. Dies ist ein sogenanntes 'self terminating command': Nach einem Zeichen kehrt es selbst wieder in den Command-Mode zurück und braucht also kein abschliessendes [ESC]. J (join) setzt die nächste Zeile ans Ende der aktuellen an.

Text löschen:
Wer viel schreibt, wird ab und zu auch mal löschen müssen. Mit **x** wird **das Zeichen unter dem Cursor gelöscht, d löscht eine ganze Zeile,** D bis ans Zeilenende. d gefolgt von einem Cursorpositionierkommando löscht die entsprechende Einheit: zB. eliminiert dw ein Wort, d$ tut dasselbe wie D, d% auf einer

Klammer eingegeben löscht alles bis zum Gegenstück der Klammer (so kannst du sehr einfach eine ganze C-Prozedur löschen...). **Grosse Textstücke lassen sich am einfachsten mit :*m*,*n*d (gefolgt von [CR]) löschen**; *m* und *n* sind Zeilennummern (herausgefunden mit Ctrl-g), $ für die letzte Zeile oder . für die aktuelle Zeile. :36,1447d löscht also Zeilen 36 bis 1447, :.,$d löscht von der aktuellen Zeile bis ans Textende.
Der gelöschte Text verschwindet aber nicht einfach im Nirwana, sondern wird in Puffern gespeichert. **p setzt den Inhalt des letzten Puffers nach dem Cursor ein**, P vor dem Cursor. Wo ein 'cut' (dd) und ein 'paste' (p) ist, kann ein 'copy' nicht weit sein; tatsächlich lässt sich mit **yy** (yank) **die aktuelle Zeile in den Puffer übernehmen**, ohne dass der Text verändert wird.
Ein Geheimtip: Du hast bis jetzt sicher bemerkt, dass viele Buchstaben ein unterschiedliches Kommando aufrufen, je nachdem, ob sie gross oder klein geschrieben werden. **Wenn sich vi also mal seltsam benehmen sollte** und kein Kommando mehr tut, was du eigentlich erwartest, hast du wahrscheinlich **versehentlich Caps-Lock gedrückt**...

Befehle wiederholen oder rückgängig machen:
Fast jeder **Befehl lässt sich durch Voranstellen einer Zahl mehrfach ausführen**; 5dd löscht also 5 Zeilen, 8x löscht 8 Zeichen usw. Lass mutig die Fantasie walten: 9yy nimmt 9 Zeilen in den Puffer, 4J setzt 4 Zeilen zusammen, 3r überschreibt 3 Buchstaben usw. – ganz einfach! Sehr wichtig ist auch das Kommando ., mit dem **der letzte Editierbefehl wiederholt** wird. Wenn du also vorhin eine Zeile mit dd gelöscht hast, wird mit . wieder eine Zeile gelöscht, selbst wenn du dazwischen Positionierbefehle benutzt hast (Cursor bewegt, gescrollt usw.).
Wenn bei aller Vorsicht mal was schief gegangen ist, nicht verzweifeln: **u** (undo) **macht das letzte Editierkommando rückgängig, U stellt den ursprünglichen Zustand der aktuellen Zeile wieder her** (nützt natürlich nur etwas, falls du unterdessen die vermurkste Zeile nicht verlassen hast...). Und wenn's ganz schlimm kommt, kannst du immer noch vi verlassen, ohne zu speichern: :q! eingeben (siehe auch weiter unten).

Suchen und Ersetzen:
Kein Texteditor ohne Kommandi zum Suchen und Ersetzen: **/*string* sucht ab Cursorposition vorwärts nach *string*,** ?*string* tut dasselbe rückwärts. Dabei ist *string* ein regulärer Ausdruck, dh. kann Wildcards und bestimmte Sonderzeichen enthalten. Das ist eine Wissenschaft für sich; ich komme mit drei Zeichen eigentlich gut aus: ^*string* sucht *string* am Zeilenanfang, *string*$ am Zeilenende; . steht für ein beliebiges Zeichen. Mit einem vorangestellten \ verlieren die Sonderzeichen ihre Spezialbedeutung, dh. um 'US$ 25.-' zu suchen, musst du für *string* 'US\$ 25\.-' eingeben. **n wiederholt den letzten Suchbefehl**, N tut dies in die umkehrte Richtung. In vi gibt es zwar ein sehr flexibles und komfortables Kommando zum Ersetzen, aber falls du nur wenige Ersetzungen durchführen musst und erst noch jede einzelne bestätigen möchtest, bist du eigentlich mit den bisher besprochenen Kommandi schon gerüstet. Keine Ahnung, wie das gehen soll? Ganz einfach: **um *wort1* durch *wort2* zu ersetzen, suchst du zuerst *wort1* (mit /*wort1*) und ersetzt es dann mit cw*wort2* gefolgt von ESC. Mit n suchst du wieder, mit . wird wieder ersetzt (falls du möchtest) usw.** So einfach, dass auch ich manchmal zu faul bin, das

folgende Kommando zu bemühen: *:n,ms /str1/str2/opt* ersetzt ab Zeile *n* bis Zeile *m string1* durch *string2*. In *opt* können die Buchstaben g, c und p vorkommen: g (global) bewirkt, dass alle Vorkommen in einer Zeile anstatt nur das erste ersetzt werden; c (confirm) ersetzt nur bei Bestätigung mit y, [CR] sucht weiter, ohne zu ersetzen; p (print) zeigt geänderte Zeilen an. Um im ganzen Text 'Studenten ' durch 'Studierende ' ohne Rückfrage zu ersetzen, tippst du also :s1,$s/Studenten /Studierende /g und [CR]. Das letzte :s-Kommando kannst du mit & wiederholen.

Fileoperationen:
Wie man den vi startet, hast du vermutlich schon bemerkt: **vi *file(s)* öffnet vi, um *file* zu editieren.** Du kannst sogar mehrere Files aufs Mal öffnen; zB. hast du nach vi *.sc alle entsprechenden Files offen, kannst das erste bearbeiten, speichern und dann mit :n (next) zum nächsten springen usw. Auch Zusammenfügen (merge) von Files ist möglich: **:r *file* lädt *file* und fügt es an der Cursorposition ein.** Irgendwann soll das File ja auch wieder auf die Disk: **:w speichert das File, mit :w *file*** kannst du es **unter einem andern Namen** (hier *file*) **speichern.** *:m,n*w *file* speichert die Zeilen *m* bis *n* in *file*, *:m,n*w>>*file* hängt sie an *file* an. **:wq** (oder ZZ für Schreibfaule wie mich) **speichert und verlässt vi.** Falls mal nichts mehr zu retten ist oder du versehentlich geändert hast, hilft die **'Notbremse' :q!**, sie **verlässt vi, ohne zu speichern.**

Optionen:
vi hat noch jede Menge Optionen, die man mit :set ... setzen und mit :set no... ausschalten kann. Ich brauche eigentlich nur zwei ab und zu: Mit :set list werden die Tabulatorzeichen im Text durch '^I' und die end-of-lines durch '$' dargestellt (:set nolist kehrt zur normalen Darstellung zurück); recht nützlich etwa, um festzustellen, welche Zwischenräume im Text wirklich Leerzeichen sind und welche von Tabulatoren kommen. Der zweite Befehl hat ebenfalls mit Tabulatoren zu tun: :set ts=4 setzt die Tabulatorweite auf 4 (default ist 8), was Listings von allzutief verschachteltem Programmcode manchmal leserlicher macht (allerdings nur, wenn der Schreiber nie Spaces zum Einrücken verwendet hat...).

Edith Birrer, IIIC/7

Beachtet dazu auch die gleich nachfolgende Befehlsübersicht! Red.

Zum Schluss nochmals eine **Kurzübersicht:**

Allgemeines:

[ESC]	**wechselt vom Insert-Mode in den Command-Mode**
Ctrl-g	**zeigt Filename, Zeilennummer, Anzahl Zeilen**

Anstelle von Zeilennummern kann auch . (für die aktuelle Zeile) oder $ (für die letzte Zeile) angegeben werden.

Wenn sich vi seltsam benimmt, sieh nach, ob du versehentlich [Caps-Lock] gedrückt hast.

Cursorbewegungen:

h, j, k, l	**Cursortasten (falls die "richtigen" nicht gehen)**
b, e; E, B	(begin/end) wortweise rückwärts/vorwärts
0 (Null)	an den Zeilenanfang
$	ans Zeilenende
Ctrl-b, Ctrl-f	(backward/forward) **seitenweise rückwärts/vorwärts**
:*n*[CR]	**zur Zeile *n***
G	auf die letzte Zeile
%	(nur auf einer Klammer:) springt zu ihrem Gegenstück

Mode-Wechsel, Text einfügen und überschreiben:

i	(insert) **Einfügen an Cursorposition**
A	(append) Einfügen am Ende der Zeile
o	(open) **Einfügen auf freier Zeile unterhalb Cursor**
O (grosses o)	Einfügen auf freie Zeile oberhalb Cursor
r*char*	(replace) 1 Zeichen ersetzen durch *char*
R*text*[ESC]	‹berschreiben mit *text*
cw*text*[ESC]	(change word) Ersetzen bis zum Wortende durch *text*
J	nächste Zeile ans Ende der aktuellen ansetzen

Text löschen, puffern, einsetzen:

(Gelöschter Text wird in Puffer geschrieben.)

x	**löscht Zeichen unter Cursor**
D	löscht bis Zeilenende
d*cursor_cmd*	löscht bis Cursorpositionierkommando *cursor_cmd*
dd	**löscht ganze Zeile**
:*m*,*n*d[CR]	**löscht Zeilen *m* bis *n***
yy	(yank) **übernimmt aktuelle Zeile in den Puffer**
p	**setzt Inhalt des letzten Puffers nach dem**

	Cursor ein
P	setzt Inhalt des letzten Puffers vor dem Cursor ein

Suchen und Ersetzen:

***/string*[CR]**	**sucht ab Cursorposition vorwärts nach *string***
?string[CR]	sucht ab Cursorposition rückwärts nach *string* (^ steht für Zeilenanfang, $ für Zeilenende, . teht für beliebiges Zeichen, \ maskiert diese Sonderzeichen)
n	**wiederholt den letzten Suchbefehl**
N	wiederholt letzten Suchbefehl in umkehrter Richtung
:n,ms/str1/str2/opt[CR]	ersetzt ab Zeile *n* bis Zeile *m str1* durch *str2; opt* kann sein: g (global), c (confirm, Bestätigung mit y, weiter mit [CR]), p (print, geänderte Zeilen anzeigen)
&	wiederholt das letzte Ersetzkommando (:s)

Befehle wiederholen oder rückgängig machen:
Durch Voranstellen einer Zahl lassen sich die meisten Befehle mehrfach ausführen.

. (Punkt)	**wiederholt den letzten Editierbefehl**
n	**wiederholt den letzten Suchbefehl**
N	wiederholt letzten Suchbefehl in umkehrte Richtung
&	wiederholt das letzte Ersetzkommando (:s)
u	(undo) **macht letztes Editierkommando rückgängig**
U	**aktuelle Zeile in den ursprünglichen Zustand**

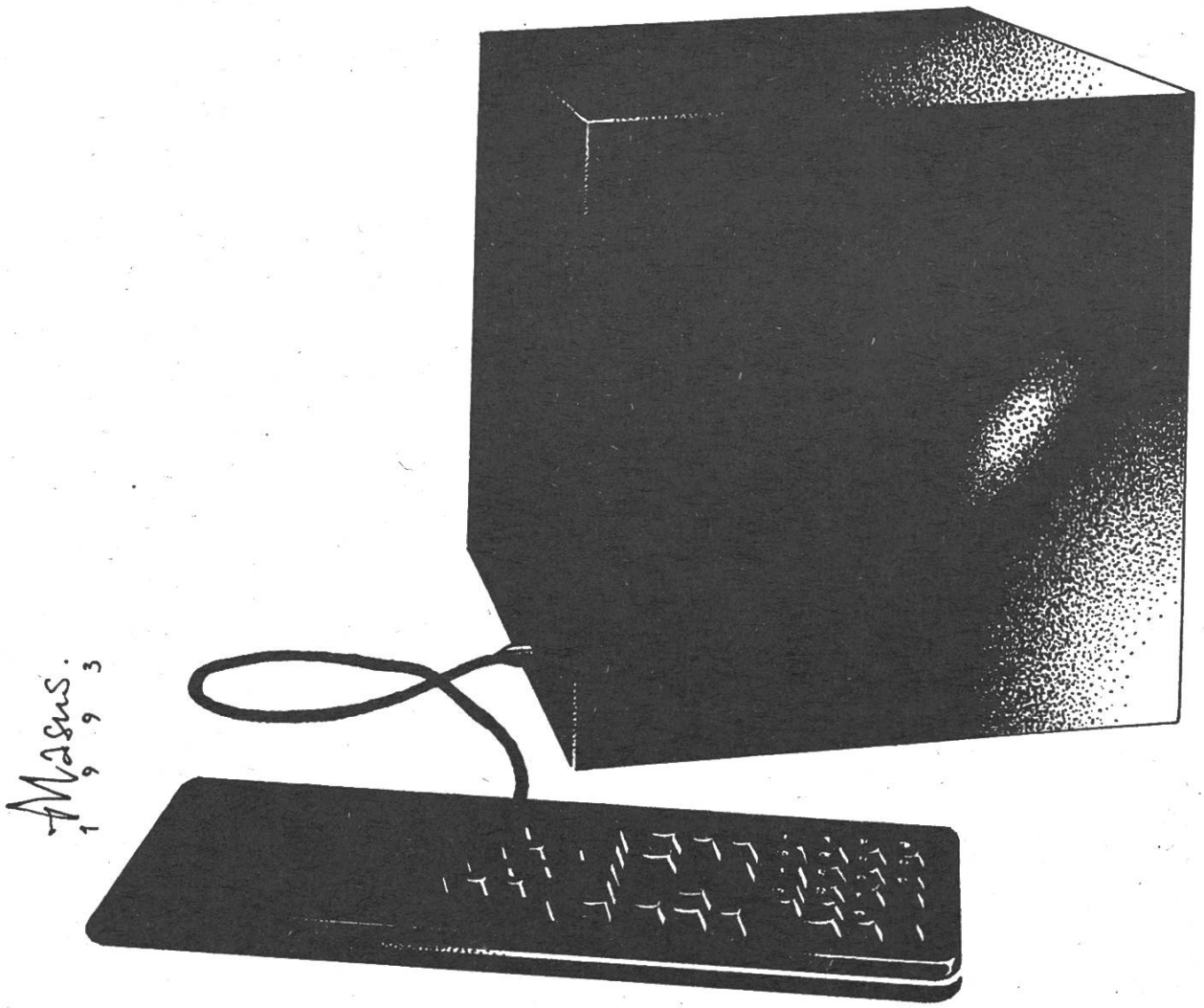
Fileoperationen:

vi *file(s)*[CR]	**öffnet vi, um *file* zu editieren** (auch mehrere Files)
:n[CR]	(next) nächstes File (falls mehrere gleichzeitig geöffnet)
:r *file*[CR]	**fügt *file* an Cursorposition ein**
:w[CR]	**speichert aktuelles File**
:w *file*[CR]	**speichert aktuelles File als *file***
*:m,n*w>>*file*[CR]	speichert Zeilen *m* bis *n* in *file*
*:m,n*w>>*file*[CR]	hängt Zeilen *m* bis *n* an *file* an
:wq[CR] (oder ZZ)	**speichert und verlässt vi**
:q![CR]	**verlässt vi, ohne zu speichern**

Optionen:

:set list[CR]	Tabulatorzeichen als '^I', end-of-lines als '$' darstellen
:set nolist[CR]	kehrt zur normalen Darstellung zurück
:set ts=4[CR]	setzt Tabulatorweite auf 4 (default 8)

Spielzeuge der InformatikerInnen, 6. Folge



Die Black Box

Wir sind ein erfolgreiches international tätiges Unternehmen mit über 10'000 Mitarbeitern. **Unser PC-Entwicklerteam hat zusätzliche interessante Aufgaben übernommen und braucht Verstärkung.** Wir suchen deshalb eine(n) fachlich ausgewiesene(n)

System-Entwickler/in

In dieser Funktion entwickeln Sie neue Applikationen für grafische Benutzeroberflächen. Ihr Verantwortungsbereich erstreckt sich zudem auf die Pflege und Integration von dezentralen Datenbanksystemen. Ferner sind Sie mit der Administration und Weiterführung der Entwicklungsumgebungen betraut.

Für diese verantwortungsvolle Position richten wir uns an belastbare und teamfähige Persönlichkeiten. Erfahrungen bei der Softwareentwicklung unter MS-Windows mit C/C++ und beim Einsatz von ORACLE, sind weitere Voraussetzungen.

Eine höhere Fachausbildung (z.B. HWV, HTL, Eidg. Diplom) ist von Vorteil. Idealalter: 25–35 Jahre.

Wenn Sie sich für diese herausfordernde Aufgabe interessieren, bitten wir Sie um Zustellung der üblichen Unterlagen an

Basler, Versicherungs-Gesellschaft
Personalwesen, Frau M. Hert
Aeschengraben 21, 4002 Basel

Basler
Versicherungen

Syst.-Entw.

Neues aus der Abteilung für Informatik

Die Prüfungssession ist vorbei und somit bleibt dem Abteilungssekretariat etwas mehr Zeit für die Implementierung des neuen Studienplanes 93:

Verzeichnis der Lehrveranstaltungen
Ab sofort ist die neue Standard-Nebenfach-Liste für Studierende nach dem neuen Studienplan 93 auf dem Abteilungssekretariat erhältlich. Auch die Verzeichnisse der Lehrveranstaltungen für das laufende Wintersemester und das nächste Sommersemester können dort bezogen werden. Diese enthalten insbesondere die Angaben über die Krediteinheiten (s.u.), die pro Vorlesung gutgeschrieben werden und ob die Vorlesungen überhaupt anerkannt werden. Aus terminlichen Gründen (das Semesterprogramm wurde schon im Juni gedruckt, der neue Studienplan erst Ende September verabschiedet) konnten Inkonsistenzen zwischen Semesterprogramm und Verzeichnis der Lehrveranstaltungen nicht vermieden werden. In Zukunft wird das Semesterprogramm dem neuen Studienplan 93 besser entsprechen.

Zuteilung der Krediteinheiten
Bei der allgemeinen Zuteilung der Krediteinheiten hat sich folgendes geändert. Neu werden Übungen weniger stark gewichtet. Somit werden alle Lehrveranstaltungen vom Typus nV+mU mit 2*n+2 Krediteinheiten bewertet. Stunden vom Typ G werden per default zur Hälfte als Vorlesungen betrachtet, der Rest ist Übungen. Verbindlich sind jedoch die Angaben im obenerwähnten Verzeichnis der Lehrveranstaltungen.

Kernfach Wissenschaftliches Rechnen
Für die Studierenden nach SP93 wird ab Wintersemester 94/95 ein fünftes Kernfach im Bereich Wissenschaftliches Rechnen zur Verfügung stehen. Der Inhalt und allenfalls notwendige Änderungen im Grundstudium werden im Moment noch diskutiert.

Lehrevaluation durch das DiZ
Im Auftrag der Schulleitung und der Abteilung evaluiert das Didaktische Zentrum der ETH in diesem Semester die Vorlesungen an unserer Abteilung durch Befragung der Studierenden mit Fragebögen. Die Details und das Verfahren werden später bekanntgegeben.
Übrigens wird, durch eine externe Expertenkommission, auch die Forschung am Departement Informatik im nächsten Frühling evaluiert.

Mentorsystem
Unterdessen habt Ihr sicher schon mitbekommen, dass Ihr Euer Fachstudium mit einem Professor diskutieren solltet, bevor Ihr Euch in die Prüfungen stürzt. Falls Ihr Euch nicht schon Gedanken gemacht habt, sollt Ihr bei Gelegenheit mal überlegen, in welche fachliche Richtung Ihr Euer Studium lenken wollt. Das hat nämlich auch Einfluss auf die Wahl des Mentors.

Prüfungen F94
Die nächste Prüfungssession wird schon fleissig vorbereitet. An der letzten AK wurden zum Beispiel schon die Prüfungsmodi festgelegt:
Die Vordiplome werden im wesentlichen von denselben Examinatoren wie diesen Herbst geprüft. Auch der Prüfungsstoff ist derselbe.
Auch die Kernfächer D&R und R&K werden vom selben Dozenten auf die gleiche Art und mit demselben Stoff wie im Herbst geprüft. Für die Kernfächer SS und IS wird es zwei Varianten der (schriftlichen) Prüfung geben: Eine basierend auf dem Stoff des letzten Wintersemesters 92/93 und eine basierend auf dem Stoff dieses Wintersemesters 93/94.
Für Compilerbau I werden getrennte Prüfungen von Prof. Mössenböck und Prof. Wirth für jeweils Ihre Vorlesung stattfinden. Beide Professoren prüfen mündlich. In der Vertiefung wird ansonsten OOP und Neuronale Netze schriftlich geprüft, die anderen mündlich.
Wie immer sind natürlich nur die Angaben auf dem persönlichen Prüfungsplan verbindlich.

Urlaubsemester
Für Studierende nach dem neuen Studienplan zählt das Rektorat bei Urlaubssemestern im Fachstudium die Semesterzahlen durchgehend hoch. Wer also nach dem 4. Semester im Winter ein Urlaubssemester macht, der ist im nächsten Sommer im 6. Semester. Die Beschränkung auf 4 Jahre im Fachstudium ist aber extra so hoch angesetzt, dass auch bei zwei Urlaubssemestern die normalen 4 Semester Fachstudium, die Diplomarbeit und sogar ein weiteres Reservesemester für Notfälle Platz haben.
Wichtig ist auch, dass es mit dem neuen Studienplan nicht mehr möglich ist, Urlaub mit Fächerbelegung zu nehmen, weil Prüfungen nur gemacht werden können, wenn man regulär eingeschrieben ist.

SATW-Stipendien
Die Schweizerische Akademie der Technischen Wissenschaften bietet Absolventen eines Ingenieurstudiums (Alter bis 35 Jahre) wenn möglich mit 2jähriger Erfahrung in praxisbezogenen Projekten die Möglichkeit Auslandserfahrung (z.B. in der GUS, Vorbedingung: Russischkenntnisse) zu sammeln. Weitere Auskünfte sind auf dem Abteilungssekretariat zu erhalten.

Mentorensystem im Fachstudium

Im Zusammenhang mit der Verwirklichung des neuen Studienplanes wird als Dienstleistung an die Studierenden ein Mentorensystem eingefuehrt. Als Mentoren wirken ausschliesslich Informatikprofessoren.
Die grundsätzliche Idee besteht darin, dass jede(r) Studierende zu Beginn seines Fachstudiums (ab 5. Semester) einen seiner geplanten Hauptrichtung nahestehenden Mentor aufsucht und mit ihm in der Folge semesterweise seinen Plan zur Belegung von Lehrveranstaltungen bespricht. Das Hauptziel dabei ist die Vermeidung von Doppelspurigkeiten und grösseren Luecken im Studiengang.

Die Einschreibebogen der einzelnen Mentoren liegen im Abteilungssekretariat auf.

Jürg Gutknecht, Abteilungsvorsteher

Das Lächeln der Sphinx[5]

Ha! Hast Du gedacht, Du könntest hier ewig als Hohepriester der Göttin des Wachstums[6] dem König der Elfen[7] huldigen? Ja, ja, siehst Du, da stehst Du nun im dritten Jahr Deines Lebens an dieser altehrwürdigen Schule und denkst Dir wohl nichts Böses. Doch heimlich, still und leise bereitet sich die Sphinx darauf vor, das grausamste, brutalste und hoffnungsloseste aller Rätsel zu stellen: "C?". Dieses fürchterlich unmenschliche Wortgebilde, vor dem die gesamte bisher gekannte Abteilung immer in die Knie ging, stellt sich gerade Dir in den Weg. Und kein Davonlaufen und kein Davonkommen – Du musst da durch!

O fürchterlich-schreckliches, unbarmherziges Alptraumszenario!

Dir in diesen ach so schweren Stunden mit Rat beizustehen, haben sich jetzt ein paar kräftige, tollkühne, allvis-sende Recken aufgemacht (unter ihnen ein neuer Οιδιπος[8] gar?). Sie werden Dir in den nächsten Ausgaben helfen, dieses Rätsel zu verstehen und zu begreifen. Damit Du in der Stunde aller Stunden der Sphinx mit hocherhobenem Haupt gegenübertreten kannst um ihr zu antworten:

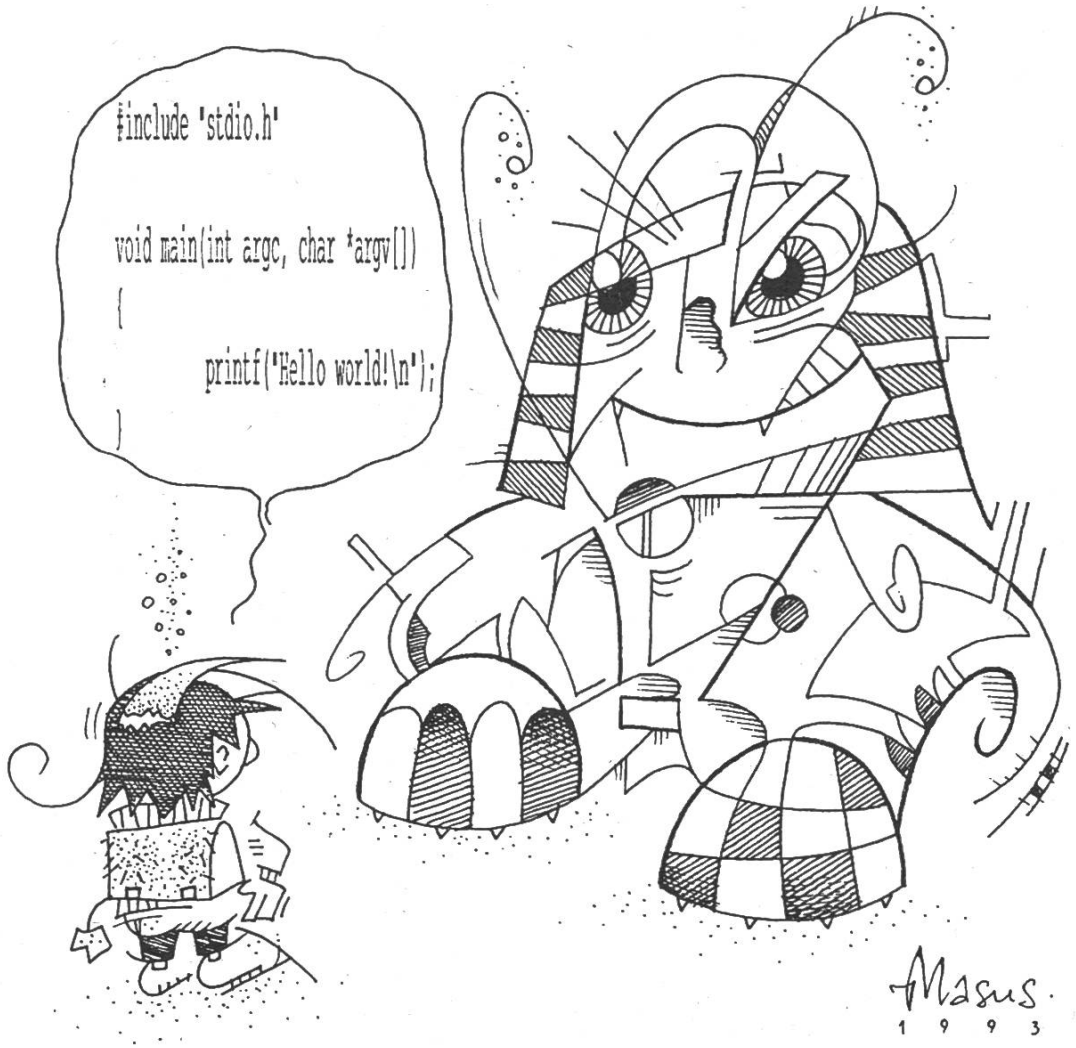
---

5 Sphinx= grausames Tier (Löwenkörper, Frauenkopf) der Antike, das beinahe unlösbare Rätsel stellt und die Nichtwissenden verzehrt.

6 Ceres = römische Göttin des Wachstums

7 Oberon= König der Elfen (in Shakespeare's Sommernachtstraum)

8 Οιδιπος (Ödipus)= sagenhafter griech. König, der seinen Vater im Kampf erschlägt, das Rätsel der Sphinx löst und die eigene Mutter heiratet.



Wovon kündet uns diese Mär? Was will uns diese Sage lehren?

Nun, so sei's denn gesagt: Der VIS will Dir *C* beibringen. Denn wer ab fünftem Semester kein C kann, der wird an einen Felsen gekettet, und jeden Tag kommt ein Adler und frisst ihm seine Eingeweide aus dem Leibe, auf dass sie bis morgen nachwachsen[9] – aber das ist eine andere Geschichte und soll ein andermal erzählt werden[10].

"So nicht!", haben wir uns gesagt, dieser Stall des Augias muss ausgemistet[11] werden und als Reinigungstool brauchen wir unseren **C-Kurs**.

Also: Kurze knappe Lektionen mit ein paar Aufgaben sollen Dir helfen, die lingua franca der Computerei zu erlernen. Und damit Du in Babylon[12] nicht verzweifelst, gehen wir davon aus, dass Du PASCAL, Modula-2 oder Oberon(-2) kannst. Ein bisschen zumindest. lj, pal

9 Prometheus, der nach der griech. Sage den Menschen das Feuer brachte, erlitt dieses Schicksal.

10 siehe bei Michael Ende, sonst wird diese Geschichte unendlich.

11 Das ist eine Arbeit für einen wahren Helden wie Herakles – und für uns Visionäre erst recht.

12 Die babylonische Sprachverwirrung: Gen, 11 (Bibel, AT)

Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

INFORMATIONSVERANSTALTUNG

für Studierende der ETH Zürich über

das europäische Programm zur Förderung von Student/innenmobilität

ERASMUS

EuRopean Community **A**ction **S**cheme for the **M**obility of **U**niversity **S**tudents

am **Freitag, 10. Dez. 1993**

von **12.15-13.00 Uhr**

im **HG E7** Hauptgebäude, Rämistrasse 101

Mitwirkung: B. Fejfar, Büro ERASMUS Schweiz; M. Bächli, Mobilitätsstelle ETH

ERASMUS bietet seit dem akademischen Jahr 1992/93 den Studierenden der ETH die Möglichkeit, einen anerkannten Teil ihres Studiums an einer ausländischen Universität zu absolvieren. Entsprechende Mobilitätsstipendien stehen zur Verfügung.

Informationen bei: Mobilitätsstelle ETH, Frau M. Bächli, HG F68.4, Tel. 632 2352,
Oeffnungszeiten: Mo, Mi, Do 13.30-17.00, oder nach Vereinbarung.

Antritts- und Abschiedsvorlesungen

Auch dieses Semester finden an der ETH wieder einige Antritts- und Abschiedsvorlesungen statt. Diejenigen darunter, die mir (pal) irgendwie für den Durchschnittsinformatikstudi interessant erschienen, möchte ich hier auflisten. Wer mit meinem Verstandnis für *interessant* nichts anfangen kann, den muss ich an die Aushänge verweisen, wo die Vorlesungen auf weissen und roten Kärtchen kurz erwaehnt sind. Die Tips fürs nächste Jahr folgen in der naechsten Ausgabe. Übrigens: Viele dieser Vorlesungen sind mit einer gewissen Allgemeinbildung durchaus verständlich. Es lohnt sich also, dort einmal hereinzuschauen.

Prof. Dr. U. Keller, "Revolution in der Erzeugung ultrakurzer Laserpulse", 18. Nov. 1993

Prof. Dr. M. Bronstein, "Computer + Algebra = Fortschritt in der Mathematik", 29. Nov. 1993

PD Dr. C. Klüppelberg, "Katastophen: Modellierung und Vorhersage", 1. Dez. 1993

Prof. Dr. J. W. Blatter, "Hochtemperatur-Supraleitung: Von den Grundlagen zur Technologie", 8. Dez. 1993

PD Dr. S. K. Solanki, "Ein leuchtendes Beispiel einer magnetischen Persönlichkeit: Die Sonne", 14. Dez. 1993

Alle *genannten* Veranstaltungen finden an den entsprechenden Tagen um 17.15 Uhr im Auditorium Maximum statt (HG F-Stock).

pal

Vertiefung Hochleistungsrechner: Architektur und Programmierung sowie Algorithmen und Anwendungen

Dozent: J. Halin

Um es gleich vorweg zu nehmen: Es ist erstaunlich, wie der Autor gewisse Tatsachen besonders negativ auslegen kann, oder auch ganz einfach falsche Aussagen in seinem Artikel macht.

Im folgenden möchte ich nun auf einzelne Punkte des Artikels eingehen.

1. ...sie wüssten, was eine Folienshow sei,... ...Der Student wird in der Vorlesung mit gegen 600 verschiedenen Folien konfrontiert... ...Dies wäre alles noch einigermassen tragbar, wenn den Studenten Kopien der Folien vorliegen würden...

Als erstes muss die Zahl 600 auf etwa 250 gekürzt werden. Das heisst also, dass man pro zweistündiger Vorlesung etwa 20 Folien gesehen hat. Natürlich ist das viel, aber es werden auch viele Rechner und -Architekturen besprochen. Und wenn man begriffen hat, was ein Vektorrechner ist, dann brauch man auch nicht mehr viel Zeit, um die Architektur eines anderen Herstellers zu verstehen. Vektorrechner bleibt Vektorrechner. Die Aussage, dass den Studenten keine Folienkopien zur Verfügung standen, liegt wohl eher an den Studenten als am Dozenten. Denn schon in der ersten oder zweiten Stunde hat Herr Halin nach einem "freiwilligen" gesucht, der die Folien für die Studentinnen und Studenten kopiert. Es hat sich leider niemand gemeldet – auch der Autor des letzten Artikels nicht. Und so hatten die Folien eben einige Verspätung.

2. Die Übungen waren leider auch nicht besser organisiert. ... Der einzige Assistent, wenn er überhaupt auftaucht...

Dazu ist zu sagen, dass der Assistent der Vorlesung *Hochleistungsrechner: Architektur und Programmierung* ***immer*** anwesend war. Ausserdem wusste er, wie die Compiler heissen und wie man sich einloggt. Denn er arbeitet schliesslich selber auf diversen Rechnern. Ausserdem läuft auf den Rechnen – mit Ausnahme der Cray – ein ganz normales, interaktives Unix wie auf den Sun-Maschinen. Und dass die Passwörter nicht durch den Raum gerufen werden, ist ja wohl klar. Ausserdem konnte man sich jederzeit beim Assistenten ein Manual für den jeweiligen Compiler ausleihen, auch für längere Zeit.
Der Autor hat hier den grossen Fehler gemacht, die Übungen einer anderen Vorlesung und vor allem eines anderen Assistenten zu kritisieren. Er

sprach von der Vorlesung *Hochleistungsrechner: Algorithmen und Anwendungen*. Diese hat sehr unter dem betreuenden Assistenten gelitten, da er zum Teil nicht in den Übungen erschien und zum anderen die Konti für die Rechner nicht organisiert hatte. Doch im Artikel der letzten Visionen ging es ja um *Hochleistungsrechner: Architektur und Programmierung*.

3. ... es ist von Vorteil, wenn man ein gewisses Wissen über Fortran mitbringt ...

Ganz bestimmt ist dies von Vorteil. Aber keinesfalls notwendig. Denn jeder, der schon mal ein programm in einer höheren Programmiersprache wie z.B c oder Pascal geschrieben hat, der kann auch in Fortran programmieren. Die vielen Vorteile, welche die in den HLR-Vorlesungen verwendete Sprache Fortran bietet, müssen gar nicht alle bekannt sein, um die Übungen bearbeiten zu können. Ein kleines Beispielprogramm wie ein "hello world" reicht im Prinzip, und dann kann man das wichtigste.

3. Geschrieben werden die Programme zuerst auf der Sun, damit man sich mit Fortran vertraut machen kann.

Dass die Programme zuerst auf der Sun geschrieben werden, hat weniger damit zu tun, dass man sich mit Fortran vertraut macht, sondern mehr damit, dass man die (teure) cpu-Zeit der Hochleistungsrechner nicht mit dem Auffinden von Syntaxfehlern im Programm verschwendet. Ausserdem wäre es unmöglich, auf der Cray einen Text zu editieren, da es nur Batch-jobs auf diesem Rechner gibt.

5. ... dass die Übungen nicht vorbesprochen und auch nicht korrigiert werden...

Nun, das mit der Vorbesprechung ist meiner Meinung nach auch nicht unbedingt nötig, da (hoffentlich) jeder Studierende *weiss*, wie ein Quicksort aussieht und auch das Skalar-, Vektor- und Matrixprodukt kennt. Das restliche Wissen über Parallelisierung und Vektorisierung wird in der Vorlesung vermittelt.
Ausserdem weiss ich beim besten Willen nicht, wie eine Korrektur aussehen sollte; ob zwei Matrizen richtig multipliziert sind, kann man schliesslich selber recht schnell mit Matlab oder mit dem Taschenrechner nachprüfen. Und ob die Programme vektorisierbar sind, steht in den Compilerlistings (natürlich mit Begründung).
Ausserdem fand in der letzten Übungsstunde eine Art Seminar statt, wo jede Übungsgruppe Ihre gelösten Probleme vorstellte, Listings verteilte und diese dann besprochen wurden.
Und falls in den Übungen doch irgendwelche Probleme auftauchten, war Herr Halin in fast jeder Übungsstunde selber anwesend und konnte Auskunft geben.

Fazit

Wenn ein Student schon eine Vorlesung in den Visionen beurteilt, dann sollte das doch bitte in einer Art und Weise geschehen, die den Tatsachen entspricht. Vor allem sollten nicht zwei verschiedene Vorlesungen miteinander vermischt werden, wie dies im Artikel in den letzten Visionen geschehen ist. Denn dies kann ein sehr schlechtes Licht auf die betreffenden Betreuer werfen, die eigentlich gar nichts mit dieser Kritik zu tun haben. Ausserdem kann ich mir vorstellen, dass es etliche Studierende gibt, die nun in das Fachstudium eingetreten sind und von diesem Artikel abgeschreckt wurden, die Vorlesung HLR zu besuchen. Dabei sind die Vorlesung *und* die Übungen sehr interessant. Dies setzt allerdings auch das Interesse der Studierenden voraus. Ist dies nicht vorhanden, kann eben ein Text entstehen, wie er in den letzten Visionen erschienen ist.

[*Die "Rage" des Verfassers in Ehren, aber diese Artikelserie wurde gerade dazu ins Leben gerufen, dass der Schreiber so schreiben kann, wie ihm der Schnabel gewachsen ist – oder dass er, wie es Christian Franz einmal ausdrückte,* Tacheles *reden kann, "frisch von der Leber weg". Redaktion]*

EASY SCHTUTZ!

Am 9. November trafen sich im D-Stock des IFW zum ersten Mal Käufer und Verkäufer zu einem Bücherbazar. Leider waren unter den ca. 40 Teilnehmern zu wenig Verkäufer zu finden. Die Nachfrage überwog das Angebot bei weitem, was sich in vielen langen Gesichtern unter den Erst- und Drittsemestrigen widerspiegelte. Für diejenigen, welche nicht zu faul gewesen waren, ihre schon längst verstaubten alten Schinken wieder aus irgend einer 'Auf-Nimmer-Wiedersehen'-Schuhschachtel zu kramen und anzuschleppen war es aber ein voller Erfolg. Natürlich brachte man den guten Chung nicht ohne viel Überredungskunst an den Mann (die Pennäler hatten ja schliesslich den VIS-Survival-Guide genau studiert) aber Biggs & Co. gingen weg wie warme Semmeln. Dazu gab's vom VIS offerierten Kafi und Snacks, was die Enttäuschten wieder etwas aufzupäppeln vermochte. Bis zum nächsten 'Dealen' hoffe ich, dass sich das ganze auch unter den Höhersemestrigen noch etwas mehr rumspricht (easy money!!) und danke nochmals allen, welche dieses Mal mitgemacht haben.

bn

Corewars in Oberon

Corewars spielt sich im ringförmigen Speicher eines Computers ab. Mindestens zwei Programme werden in den Speicher geladen, wo sie versuchen, möglichst viele Speicherzellen zu erobern und ihre Gegner zu vernichten.

A. K. Dewdney hat dieses Spiel 1984 im "Scientific American" zum ersten Mal beschrieben. Er definierte auch die Sprache "Redcode", eine einfache Assemblersprache, in welcher die Corewarsprogramme geschrieben werden. Trotz der kleinen Zahl an Befehlen lässt die Sprache dank verschiedener Adressiermodi komplexe Manipulationen zu.

Der Reiz besteht nun darin, selbst ein starkes Programm zu schreiben und es nachher in die Kampfarena zu schicken, wo es sich gegen andere Programme behaupten muss. Die Situation im Speicher wird dann grafisch dargestellt. So ist jederzeit ersichtlich, welches Prorgramm welche Speicherzellen besitzt. Besonders spannend wird es, wenn sich Programme in mehrere Prozesse zerlegen (möglich mit dem Befehl `SPL`) und dann an verschiedenen Orten den Kampf fortführen. Eine Prozessverwaltung sorgt für eine gerechte, zyklische Abarbeitung der Programme. Programme mit vielen Prozessen werden nicht bevorteilt, denn ihre Prozesse kommen seltener zum Zug.

Der gesamte Speicher ist für alle Programme zugänglich. Bei Spielbeginn werden alle Kontrahenten an eine zufällige Startadresse geladen. Es wird keine Trennung zwischen Programm- und Datenspeicher gemacht. So können sich Programme selbst manipulieren oder den Gegner gezielt unschädlich machen (Die Arbeitsweise von einigen Computerviren.). Unschädlich macht man die Gegner dadurch, dass man ihre Prozesse "killt". Ein Programm stirbt nämlich, wenn es keinen Prozess mehr hat. Ein Prozess stirbt, wenn er versucht, eine `DAT`-Anweisung auszuführen. Ein gescheites Programm wirft also seinen Kontrahenten ein oder besser mehrere `DAT` vor die Füsse, hütet sich aber gleichzeitig, selbst darauf zu stehen, denn "wer andern eine Grube gräbt..." Speicherzellen können mit der Anweisung `MOV x y` beschrieben und damit erobert werden. Man kann so einen Gegner zum Arbeitsknecht zwingen, weil er nach geschicktem Kopieren von Speicherzellen fremden Code ausführt.

Fasziniert von Corewars, haben wir dieses als Semesterarbeit in Oberon implementiert. Nun möchten wir dieses Programm allen zugänglich machen. Wir haben deshalb im Ceresraum im IFW zwei Disketten deponiert. Eine Diskette beinhaltet das Programm, eine genaue Beschreibung der Sprache Redcode und Erklärungen zum Programm sowie einige Bei-

spielprogramme. Die zweite Diskette ist für Euch reserviert. Dorthin sollt Ihr nämlich Eure genialen, selbst erfundenen Redcodeprogramme kopieren, nachdem Ihr diese ausgiebig, gegen andere Fieslinge kämpfend, getestet habt. Wer schafft es, ein Programm zu schreiben, welches regelmässig gegen die "Mäuse" ("Mice.s", Sourcecode auf Diskette) gewinnt? Diese vermehren sich unheimlich schnell, indem sie sich unaufhörlich an andere Adressen kopieren. 1989 wurde dieses Programm Weltmeister (Es werden jedes Jahr Weltmeisterschaften ausgetragen! Letztes Jahr soll ein russisches Programm namens "Cowboy" gewonnen haben). Wie wär's, wenn eine "ETH - Program - Selection" uns den nächsten Titel sichern würde?

Nachfolgend ein paar Eigenschaften starker Programme (gewisse Eigenschaften schliessen sich gegenseitig aus) :

Dynamik:
Das Programm wandert im Speicher umher. Besser noch: es kopiert sich an verschiedene Stellen im Speicher, indem es neue Prozesse startet. Ein Verlust eines einzelnen Prozesses verursacht nur eine Schwächung, nicht den Tod.

Schutz:
Das Programm erkennt DAT - Fallen und weicht ihnen aus oder überschreibt die Zellen vor ihrer Ausführung mit harmlosen Anweisungen. Raffinierte Programme könne sich nach einem Treffer sogar selbst reparieren.

Aggressivität:
Das Programm schiesst mit DAT-Anweisungen auf seine Gegner.

Vermehrung:
Das Programm teilt sich dauernd in neue Prozesse auf, nach dem Motto "Einer wird schon irgendwie überleben.". Dadurch wird das Programm aber träge, weil die einzelnen Prozesse viel seltener an die Reihe kommen.

Schnelligkeit:
Das Programm agiert schneller als die Gegner und kann daher einer DAT - Bombardierung entgehen.

Wir hoffen, dass wir Euer Interesse geweckt haben. Kopiert Euch doch in einer stillen Minute eine Kopie unserer Diskette und probiert ein wenig herum. Vielleich kommt ein gescheites Programm raus. Lasst dann Eure Programme gegeneinander spielen...

Viel Vergügen,

Daniel Som und Roger Gatti

PS: Für Anregungen sind wir dankbar und Fragen beantworten wir auch gerne. (e-mail: djsom und rgatti)

F&K's Bieridee

Allein in Bayern gibt es über 5000 verschiedene Biersorten, an der ETH gibt es Semesterarbeiten, die sich mit Bläschen im Bier befassen, young Einstein hat das Bieratom gespalten und am Erstsemestrigenfest des VIS gibt's Freibier. Ausserdem werden auch in der Schweiz einige hundert Biere hergestellt und nach einem strengen Tag trifft man so manche Studentin oder auch manchen Studenten im "Dörfli" mit einem gelben Sirup vor sich. Und damit bin ich schon beim Thema: Bier wird von den meisten Leuten immer mit der Farbe gelb beschrieben. Dabei gibt es bei Bier eine Farbenvielfalt, wie sie sonst nur die Liköre von Bols haben. Die Farbe des Biers geht meistens von hellgelb (vor allem die Farbe desjenigen in Stangenform), über goldgelb nach braungold bis hin zu dunkelbraun und Guinness-schwarz.
Heute möchte ich nun zwei Brauereien vorstellen. Die erste steht in Arbon, genaugenommen im Restaurant Frohsinn. Die zweite ist das *Back und Brau* in Frauenfeld. Die beiden Lokale haben eines gemeinsam: Sie brauen ihr Bier selbst.
In Arbon gibt es nur ein helles und ein dunkles Bier, wie es die meisten Brauereien – sogar die Grossbetriebe – herstellen, mit einem kleinen Unterschied: Es wird strikt am Reinheitsgebot festgehalten. Man trinkt dort also ein Bier, das ohne künstliche Hilfsstoffe oder Konservierungsstoffe gebraut wird. Frisch und köstlich!
Im Back und Brau in Frauenfeld hält man sich ebenfalls an da Reinheitsgebot. Nur ist dort die Auswahl noch viel grösser. Neben dem "normalen" Hellen und Dunklen gibt es in Frauenfeld noch ein "Klosterbräu" und ein "Altbier". Es ist übrigens die einzige Brauerei in der Schweiz die mir bekannt ist, die ein richtiges Altbier braut.

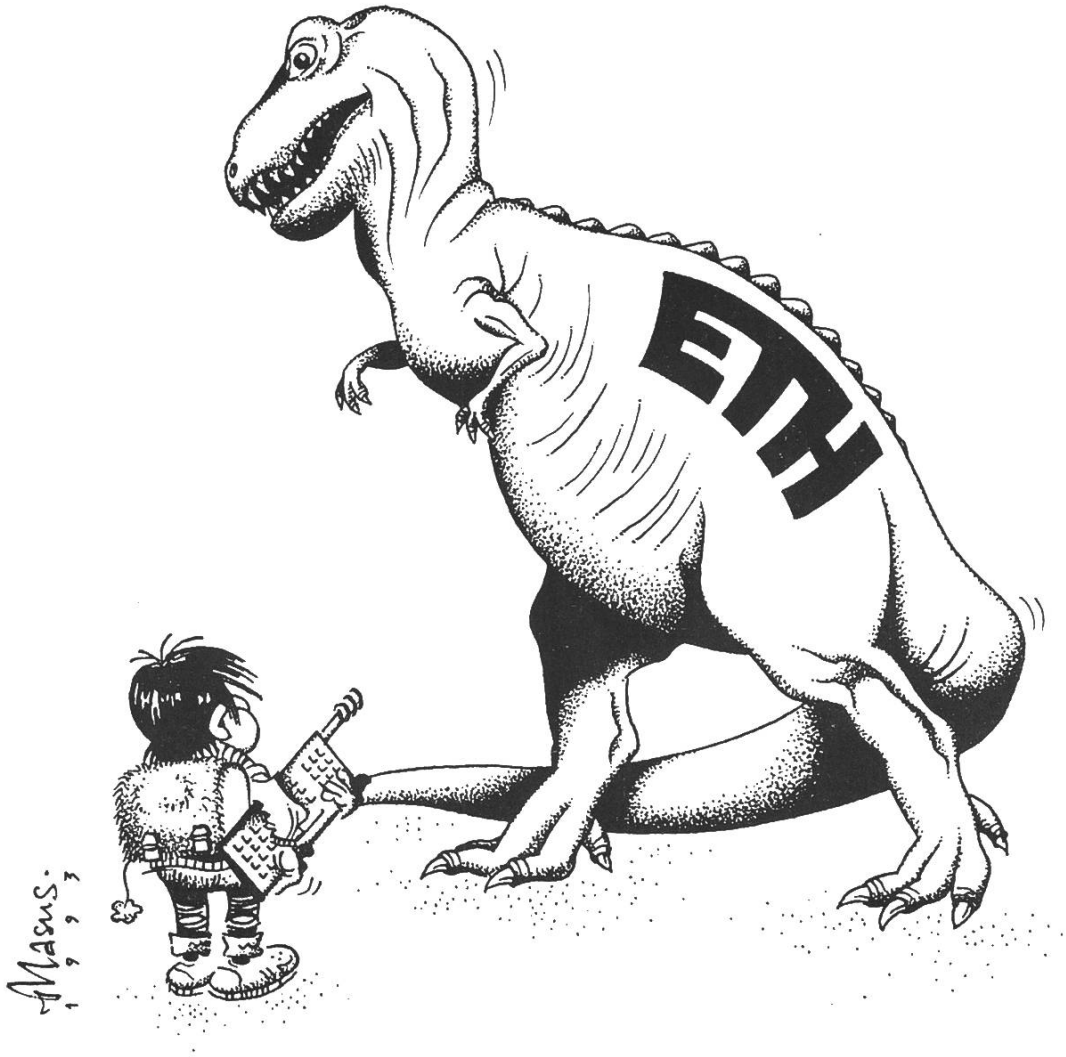
Nun, bis jetzt habe ich hier ein paar Begriffe gebraucht, die wohl einigen noch nicht bekannt sind. Aber noch ist nicht Hopfen und Malz verloren. Denn ich hoffe, dass der Redaktor auch für diese Bieridee zu haben ist und ich in den nächsten Visionen mal ein paar Begriffe wie "Altbier", "Obergärig und Untergärig" oder "Starkbier" erläutern kann.
Also bis zum nächsten Mal und Prost!

Frank

Wo bleiben die
ASCII-Cows?,
hör' ich fragen – Keine Angst, brav bleiben, und dann vielleicht nächstes Mal... pal

Aus dem Lexikon (III)

loop, endless: see endless loop



?

Don't worry.
The Guide. From VIS.

G.A.B. 8648 Mirusio

pal

Falls unzustellbar bitte zurück an:

Verein der Informatikstudierenden
IFW B29
ETH–Zentrum

CH–8092 Zürich

Inhalt