VIERTE DIMENSION

7. Jahrgang Nr. 1 - März 1991

Kleines FORTH

Schnelles Sortieren mit FORTH

Anpassung an ANS-FORTH

Mitgliederversammlung

FORTH MAGAZIN

7,50 DM

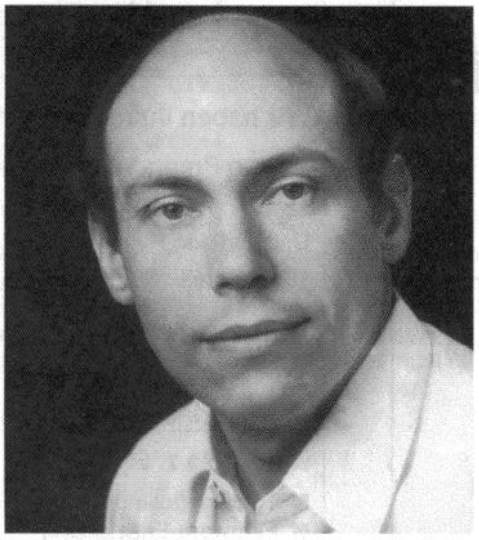
FORTH-*Magazin*
Vierte Dimension

Nr. 1 März 1991

ANZEIGEN:

BEILAGE: ***Sonderdruck, Rapid prototyping in der Geräteentwicklung , Mikrap AG/FORTH-Systeme A. Flesch***

Ein Leben voller Widersprüche



Peter Dinies, geb. am 20. März 1946

Widersprüche formten mein Leben, durch Widersprüche kam ich zur Öffentlichkeitsarbeit. Störrisch, trotzig, aufsässig ging ich schon früh meine eigenen Wege.

Zur Schule ging ich nicht gern, sie war ein lästiges Übel. Die Lehrmethoden entsprachen nicht meinem inneren Gefüge. Dennoch, ich konnte es nicht lassen, aus lauter Langeweile, bedingt durch meine vierjährige Marinezeit, das Abendgymnasium zu besuchen, um anschließend Ozeanographie und Maschinenbau zu studieren.

Mit dem Programmieren unter Zeitdruck (wie an der Uni und der Fachhochschule) hatte ich immer meine Schwierigkeiten. Die ersten erfolgreichen programmiertechnischen Gehversuche machte ich in der Entwicklungsabteilung eines großen weltweit bekannten Industrieunternehmens in Kiel. Als "Daniel Düsentrieb" arbeitete ich an Problemen, die andere nicht lösen konnten.

Bedingt durch die Firmenstrategie (Rationalisierung war angesagt), kam eine Weiterbeschäftigung nicht in Frage.

Zwischenzeitlich arbeitete ich als Butler, Gesellschafter, Chauffeur, Hauswirtschafter, Krankenpfleger, Fahrer einer Werttransportfirma, Zeitungszusteller und Kom(m)ödiant an einem Kieler Zimmertheater.

Ein 14tägiger Theateraufenthalt als Kom(m)ödiant in Göteborg (Schweden) brachte mich - in Sachen Öffentlichkeitsarbeit - auf den "rechten" Weg. Eine "Profilgestalterin" nahm mich in ihre Hände. Mein neuer Lebensweg war programmiert, ich ging zur Zeitung. Der Sprung ins kalte Wasser wurde zum Lebenselixier.

Bei einem Kieler Informations- und Anzeigenblatt erlernte ich mein Handwerk, die Kunst mit anderen Augen zu sehen, Texte "richtig" zu schreiben (das Wichtigste kommt immer zuerst) und Filme entwickeln.

Reportagen, Produktbeschreibungen, Portraits von berühmten Politikern und Wirtschaftsangehörigen folgten.

Inzwischen bin ich Mitherausgeber eines kleinen unbequemen Blattes geworden. Desweiteren betreibe ich Öffentlichkeitsarbeit für ein Sicherheitsunternehmen, bin Referent für Öffentlichkeitarbeit bei der Deutschen Lebensrettungs-Gesellschaft in Kiel, stelle Künstler bei Vernissagen vor und bin "last not least" Editor der "Vierten Dimension".

Meine Hobbies sind Fotografieren, unbekannte Kochrezepte ausprobieren und Survival in der einfachsten Form.

Peter Dinies

IMPRESSUM

FORTH-*Magazin "Vierte Dimension"*

Herausgeber:
FORTH-*Gesellschaft e.V.*

Redaktion , Satz und Layout:
Peter Dinies, Metzstraße 38, 2300 Kiel 1
Tel. 04 31/1 32 39

Redaktioneller Beirat:
Ulrich Hoffmann, Jens Storjohann, Michael Kalus

Kontaktadresse:
FORTH-*Büro,W-8044 Unterschleißheim, Postfach 1110, Tel. 0 89/3 17 37 84 oder FORTH-Mailbox, München, Tel. 0 89/7 25 96 25 8 N1 "Konferenz Vierte Dimension".*

Quelltextservice:
Der Quelltext von Beiträgen, die mit dem Diskettensymbol gekennzeichnet sind, ist auf der Leserservice-Diskette zur jeweiligen Ausgabe oder in der FORTH-*Mailbox zu finden.*

Autoren dieser Ausgabe:
Heinrich Hohl, Ulrich Hoffman, Jörg Plewe, Jörg Staben, Finn Berlev

Redaktionsschluß:
Erste Woche im mittleren Quartalsmonat. Erscheinungsweise vierteljährlich

Auflage: *Ca. 1.000*

Druck:
Buch- und Offsetdruckerei Bickel & Söhne, Frankfurter Ring 243, 8000 München 40.

Preis:
Einzelheft DM 7,50, Abonnementpreis DM 40,-, bei Auslandsadresse DM 45,- inklusive Versandkosten.

Editorial

Liebe FORTH'lerinnen
Liebe FORTH'ler!
Begriffen wie VolksFORTH/ultraFORTH, SWAP, DROP, LOOP, wie auch 32Bit Systemen kann ich zur Zeit nur Tilden, Hurenkinder und Schusterjungen, Kerning oder gar noch ein Tagged Image File Format sowie andere typografische Begriffe entgegenstellen.

Ich bin sicher, daß wir uns trotz der Vielfalt der verschiedenen Sprachen und Begriffen verstehen. Jeder in seiner eigenen Sprache und jeder in seinem eigenen Medium. Denn wer hier schreibt, blamiert sich nicht. Und wenn, dann hilft der Dialolg, um Unklarheiten auszuräumen.

Ohne den Dialog, den guten Rat ist manches nicht zu schaffen. In solchen Fällen steht der redaktionelle Beirat - oft umstritten und heiß diskutiert zur Verfügung. Dieser berät mich "den neuen Editor" wie er in der FORTH-Fachsprache heißt, in vielen sach- und fachbezogenen Problemen.

Vor allem dann, wenn der Ratgeber mit der Situation vertraut ist, wenn er die Entwicklung eines Vereins oder Unternehmens, wie die der FORTH-Gesellschaft e.V. über Jahre verfolgt hat oder über "die erforderliche Marktkenntnis" (FORTH) verfügt, ist guter Rat oft Gold wert.

Dennoch halte ich es mit meinen neuen Ideen, wenn es um die Entwicklung neuer Unternehmensstrategien wie die der "Vierte Dimension" geht, wie ein Profi. Ich reagiere erst und überlege dann.

Denn das Bewußtsein, einfach umwerfend auszusehen, eine Erscheinung zu sein, die sämtliche Blicke auf sich zieht - welche Vereinszeitschrift möchte sie nicht besitzen - macht selbstsicher und weltoffen. In diesem Sinne bleibt nichts von mir unversucht, um das Äußere der "Vierten Dimension" und das der FORTH-Gesellschaft e.V. ins rechte Licht zu rücken.

Herzlichst
Euer Editor, Peter Dinies

FORTH Maschinen: FRP1600

Der Forth Risk Processor FRP1600 ist vom Konzept ins Stadium der Studie gekommen - und funktioniert. Das Redesign soll beginnen. Man hofft den Prozessor auf der Echtzeit'91 in Sindelfingen 11.-13. Juni bereits vorstellen zu können. Dieser Forth Risk Prozessor ist kompatibel zu den Registern des 68000 ebenso wie zu dessen Interuptstrukturen. Die gesammte Bausteinfamilie kann angeschlossen werden. Der FRP1600 zeichnet sich gegenüber dem RTX2000 durch eine besserte Interuptbehandlung aus.ksk

ANS FORTH

Martin Tracy ist zur EuroFORML Konferenz 1991 in Marienbad eingeladen. Dabei soll das ANS Forth vorgestellt werden. Man rechnet damit, das bis dahin der 'draft proposed standard' dpANS soweit gesichtet und diskutiert ist, daß das ANS FORTH dann feststeht.
Bitte vormerken.

ANS Forth,
der Stand der Dinge

Es ist soweit. Ein Amerikan National Standard (ANS) für Forth wird in Kürze vorliegen. Die Prozedur ist langwierig, seit August 1987 laufen die Bemühungen der Einigung auf ein 'draft proposed standard' (dp) der nun erreicht ist. Wer sich für weitere Einzelheiten der Geschichte interessiert kann diese in der FORTH DIMENSIONS, Volume XII, Number 5 (1990) nachlesen: X3J14 Report, Elisabeth D. Rather.

Das Komitee hatte kürzlich die Arbeiten soweit abschließen können und plant das dpANS Forth vorzulegen. Jetzt wird diese BASIS der Öffentlichkeit vorgestellt. Die BASIS können Sie für $10 plus ca. $5 für Überseeversand bestellen bei:

Forth Vendor's Group

c/o FORTH Inc.

111N.Sepulveda, Manhattan Beach, CA 90266

Extracting on Topic: Meeting
Path: majestix!unido!fauern!ira.uka.de!sol.ctr.columbia.edu!lll-winken!uunet!tut.cis.ohio-state.edu!ucbvax!MITCH.ENG.SUN.COM!wmb
From: wmb@MITCH.ENG.SUN.COM
Newsgroups: comp.lang.forth
Subject: Recent ANS Forth Meeting
Message-ID: <9102041440.AA19023@ucbvax.Berkeley.EDU>
Date: 4 Feb 91 07:46:14 GMT
Sender: daemon@ucbvax.BERKELEY.EDU
Reply-To: wmb%MITCH.ENG.SUN.COM@SCFVM.GSFC.NASA.GOV
Organization: The Internet
Tue Feb 5 11:18:48 1991

I just got back from the ANS Forth meeting in Los Angeles. Most of the action occurred in the extension wordsets. The technical issues have been pretty much put to rest.

Some highlights:
New control flow words:
AHEAD (unconditional forward branch), SO (ROLL for control flow tokens), STILL (PICK for control flow tokens). These words complete the set of control flow primitives, allowing the creation of user-defined control structures to implement *any* control flow graph.

Clarification of input stream mechanism:
The new variable BLOCK-FID contains the fileid of the file containing the blocks accessible via BLOCK, BUFFER, and LOAD. If BLOCK-FID is zero, the blocks come from the system default block space (which may or may not be a file). This does not affect interpretation from text files; that still depends on the SOURCE-FILE value (-1 for string input, 0 for keyboard input, otherwise it's the fileid of the input text file).

Conditional compilation:
A wordset for conditional compilation is going out for letter ballot. It looks it may pass. The proposal calls for fully-nestable, immediate words IF-TRUE OTHERWISE IFEND (same names as in Forth 83).

Multitasking:
All mention of multitasking has been moved to an appendix that describes the "traditions" of Forth multitasking.

Decompiler:
New word SEE (decompiler) added to the toolkit extension wordset. The decompilation format is system-dependent.

EMIT?:
EMIT? (check for output device ready) added to toolkit extension wordset.

Search order:
ADDITIONS changed back to DEFINITIONS, FORTH-WORDLIST added to search order wordset, ISOLATE removed. The search order wordset now contains a precisely-defined "primitive" set of search order words, suitable for use as tools to implement any search order scheme. The search order extension wordset now contains ONLY ALSO etc.

AT-XY:
Popular sentiment suggests that the name COLROW sucks, so the name was changed to "AT-XY" (just AT was suggested, but that name has inconsistent common usage).

GET-DATE:
(or maybe it's GET-TIME ??) Returns the "wall clock" time as
(– sec min hrs day mon year)
sec:0-59 min:0-59 hrs:0-23 day:1-31 mon:1-12 year:e.g.1991
(in CORE EXT wordset).

That's all I remember right now.

Mitch Bradley

YERK

Die amerikanische Firma Kriya Systems hat sich dazu entschlossen, ihr ehemaliges Produkt NEON (Mischung zwischen Smalltalk und Forth auf dem Apple MacIntosh; Vorläufer von Actor für Windows) in public domain zu geben. Es wird jetzt unter dem Namen YERK vom Yerkes Observatorium in Chicago zu Verfügung gestellt.

7th euroFORML conference on the FORTH programming language and FORTH processors
Held from October 11th through 13rd 1991 at the Casino of Marinsk Lzn (Marienbad) Czechoslovakia
In cooperation with Forth Interest Group Inc, USA

Invitation & Call for Papers euroFORML is an international meeting of computer practitioners using FORTH as a problem solving tool. Lectures, workshops and presentations will be conducted to demonstrate techniques and problem solving strategies that have proved useful. This years conference will again focus on FORTH in real time applications and its potential in the area of robotics and industrial control applications. At euroFORML'89 for the first time participants from East-European countries participated and presented some very interesting work. FORTH is beeing used in Leningrad (USSR), Tartu (Estonia), Sofia (Bulgaria) as well as in former East-Germany and other East-European countries. Given the current political circumstances and the non-convertibility of their currencies participation for East-Europeans became even more difficult than two years ago. This is why we decided to hold this years conference at a central location in Eastern-Europe. Marinsk Lazn is located mid-way between Nuremberg and Prague. As the worlds first spa it has always been an international meeting place besides accomodating the European nobility. The hotels certainly do not live up to western standards which all too often destroys the atmosphere. Marinsk Lazn has plenty of it. Accomodation will be in a hotel with single and twin bedrooms right next to the Casino were the meetings will be held. Organising the meeting from Hamburg under the particular economic conditions of the CSFR will be more involved then usual. Therefore we ask you to register early. The conference language will be English and FORTH, of course. The conference will be self organizing ie. there will not be a strict agenda prior to the conference. If you want to present your ideas you may choose one of the following formats:

Paper presentation You will present a paper in a 15 minute talk in front of the whole group with the possibility to get immediate feedback.

Poster presentation You will be assigned a "poster space" where you can present your ideas to a small group of people in a separate room. This is especially useful for demonstrations of hard- and software.

Workshops You may organize or participate in workshops which will be organized at the beginning of the conference, depending on demand.

An English language proceedings will be published after the conference; papers to be included in the proceedings will be handed out at the beginning of the conference.

Registration Registrations should be mailed by July 1st in order to be eligible for a rate reduction. A deposit per person will be required. There is a limited number of places available for students at a substantially reduced rate. A limited number of participants from East-Europe may register till July 1st at a special rate payable in Czechoslovakian currency. Space is limited and you are assigned on a first-come, first-serve basis. More specific information about the rates will be mailed in the next couple of weeks.

Author instructions Papers to be presented at the conference (to be included in the proceedings) will have to be mailed to the organizer no later then September 23rd 1991 in camera ready form. The format is DIN A 4 (letter size) with a margin of 2,5 cm (1.5") on all sides. Every page should have a page number and the authors name. Papers should not exceed 15 pages. Code should be accompanied by shadow screens.

For reservations and conference papers write or call euroFORML - office M.Kern, K.Schleisiek-Kern / Uhlenhorster Weg 3 / D-2000 Hamburg 76 / Germany Tel: +49 40 2296441 Fax: +49 40 2297205

MOD mit Vorzeichen

Es gibt eine Anwendung, wo ("alle paar Jahre...") ein MOD einer negativen Zahl hervorkommt, nämlich die Errechnung des Wochentag's aus dem Datum mit Hilfe der "Zeller's Congruenz". Ich habe die entsprechenden Ausgaben gerade nicht parat, aber es waren im letzten Jahrgang des "Dr. DOBB's Journal" drei Artikel, die sich mit der Wochentagsberechnung und der MOD-Funktion befassten (dort allerdings mit MODULA und TURBO-Pascal). Holger Petersen, Kiel

Anregungen und Vorschläge

Wenn ich mir den Inhalt der VD der letzten Zeit ins Gedächtnis rufe, fällt mir auf, daß sich viele Artikel mit dem inneren Wirken von FORTH-Systemen auf PCs oder Maschinen mit dem 68-Tausender Prozessor befassen.

Ich meine, daß die Diskussion über Verbindungen von Forth mit den neuen Entwicklungen der Informatikwelt in der VD aber zu kurz gekommen ist.

Hier denke ich zunächst einmal an Artikel über neue Programmierumgebungen, sowohl über Hardware, als auch über Betriebssysteme und Oberflächen.

Wenn ein AT mit einem 80286 oder einem 80386SX schon als Minimalausstattung eines Studenten gesehen wird, will ich etwas über neue Möglichkeiten der PC-Welt, die 1MB-Grenze im Speicher zu durchbrechen, wie extended oder expanded memory, protected mode, DOS extender und andere erfahren.

Bestimmt hat jemand doch auch Graphik oder Numerik in FORTH programmiert. Hier möchte ich mehr Artikel über Anwendungen der neueren Graphik-Karten/Prozessoren und Coprozessoren.

Wir brauchen Informationen über Systeme, die unter verschiedenen MS-DOS-Erweiterungen wie Windows 3.0 laufen. Dann für OS/2, Unix und seine graphischen Oberflächen.

Ein PC-FORTH, das unter Minix läuft, litte nicht mehr unter den Beschränkungen des DOS, d.h. der Adressenbereich wäre 16 MB (auf einem AT); echtes Multitasking, und ein vernünftiges Dateiensystem wären auch vorhanden.

In den letzten Jahren haben sich Workstations wie die von Sun, DEC oder HP verbreitet. Sie haben meistens einen RISC-Prozessor oder einen aus der 68-Tausender Reihe, einige MB Hauptspeicher und einen Graphik-Schirm. Sie laufen unter einem Unix-Betriebssystem und sind mit Hilfe einer (fast) einheitlichen graphischen Bediener-Oberfläche recht einfach zu bedienen.

Ich möchte Artikel über FORTH-Systeme, die mit/unter diesen neuen Betriebssystemen und Bediener-Oberflächen laufen, vielleicht auch über ein von der Sprache C aus startendes leicht portierbares System.

Warum liest man so wenig über FORTH in Netzwerken, wenn in jeder Workstation die Netzwerkfähigkeit mitgeliefert wird.

Über die Implementationen auf neuen Prozessoren, insbesondere den neuen RISC-Prozessoren, erfährt man etwas aus Amerika, aber bis jetzt gab es in der VD erst eine kurze Information über ein Forth auf dem Prozessor i860 von Intel, wo die Fähigkeiten dieses Prozessors aber noch nicht genügend ausgenutzt wurden.

Über FORTH in neuen und alten Anwendungsgebieten gibt es bestimmt eine Fülle von Material, das in Schubladen oder auf Festplatten schlummert. Im Bereich Maschinensteuerungen wird FORTH seine Eignung wohl weiter unter Beweis stellen. Wo sind die Artikel darüber? Flexible Testsysteme für elektronische oder elektromechanische Komponenten hat doch bestimmt jemand in FORTH programmiert.

FORTH zur Steuerung von Peripherie-Geräten, sei es vom Hauptrechner aus, oder über in der Peripherie tätige Prozessoren, muß sich doch auch in Artikeln finden lassen. Jeder hat doch schon die Anzeigen der Fachpresse durchmustert und sich gewundert, wie billig Geräte geworden sind, die vor wenigen Jahren noch Großfirmen oder dem Militär vorbehalten waren. Ich meine Laserdrucker, Plotter, Scanner, schnelle Modems, Meß- und Steuergeräte. Vielleicht hat sich schon jemand an OCR-Systeme oder andere Mustererkennung herangetraut.

Es muß doch auch eine Verbindung zwischen FORTH und Systemen für Satz- und Textverarbeitung geben. Seit Leo Brodies QTF (Quick Text Formater) ist es um Textverarbeitung in FORTH ruhig geworden. Hat sich nicht einmal jemand mit TEX und Metafont beschäftigt und gefunden, daß sich vieles besser in FORTH programmieren ließe, zum Beispiel ein Bildschirmtreiber.

Ich hoffe, daß nun viele Leser sich schnell an ihren Rechner setzen und aus der Arbeit und den Erfahrungen der letzten Zeit einen Entwurf für einen Artikel in der VD tippen und an die Redaktion schicken.

Wer keine Zeit hat, lange an einem Artikel zu feilen und schlichten, schickt ihn eben so, wie er aus der Tastatur klackert, und läßt ihn von der Redaktion überarbeiten.

Jens Storjohann, Hamburg

LOGO, LEGO, FORTH

Was 4TH von LEGO lernen kann, wodurch LEGO noch lange nicht überflüssig wird.

Der Titel knüpft zwar an einen früheren Beitrag [1] an, aber: Nein! -es ist kein Tippfehler.Sondern er weist zurück in eine Zeit, da bestand Spielzeug noch zu großen Teilen aus Blech und ein "neues Baukastensystem" machte Furore: LEGO. Der "LEGO-Stein" als Grundelement war "rechtekkig" und hatte acht Zapfen zum Aufeinanderstecken; kleinere hatten vier Zapfen - die"Vierer" und konsequenterweise gab es dann noch "Zweier" und "Einer". Einfach, genial und flexibel. FORTH soll ja so ähnlich sein...

Aus diesen Grundelementen konnte man dann sehr vieles aufbauen, große Häuser, kleine Häuser und auch Autos; die sahen dann aus, wie kleine Häuser auf Rädern. Um eine Säule für einen Balkon zu bauen, steckte man vier "Einer" zu einer kleinen rechteckigen Stange aufeinander. Die Einer hatten zwar für einige Zeit ihre Flexibilität eingebüßt, auch als Schornsteine oder Burgzinnen dienen zu können, aber später konnte man sie wieder umfunktionieren. Daraus wurden dann Schiffe zusammengebaut, diese wieder zerlegt und in Lastwagen verwandelt. So konnte man einige wenige Grundelemente immer wieder neu kombinieren. Besaß man aber einen großen Haufen LEGO-Steine, neigte man nach einiger Zeit dazu, solche Säulen aus "Einern", die man doch immer wieder brauchte, nicht mehr zu zerlegen. Barbaren sollen sogar LEGO-Steine zu Mauern verklebt haben

und hatten damit ein Haus ruck-zuck fertig! Alles getreu der Devise: Schneller zum Ziel. Mittlerweile haben in einem LEGO-Baukasten nur noch etwa zwei Drittel der Bausteine eine allgemeine Funktion; das ganze restliche Drittel der Bau-Elemente dagegen sind Spezialbausteine geworden: Glaskuppeln für Raumschiffe, Gelenke für Schwenkflügel-Flugzeuge, Schienen für Zahnradbahnen und - rechteckige Stangen, die genau so hoch sind wie vier "Einer". Die LEGO-Modelle sehen jetzt durch diese Spezial-Elemente auch aus wie "richtige" Zahradbahnen, Raumstationen und Ritterburgen. Deshalb halte ich das Programmiersystem für das beste, das leistungsfähige Spezialfunktionen ebenso wie allgemein verwendbare Funktionen bietet. Schon wäre es, wenn ein solches Programmiersystem dann ein FORTH-System wäre.

P.S.:
LEGO ist bestimmt [c] LEGO oder so, auf jeden Fall ein eingetragenes Warenzeichen und wer nachgemachte oder verfälschte LEGO-Steine verbaut oder ..., wird mit Fischer-Technik [c] nicht unter ... bestraft.

Quellen:
[1] Martin Holzapfel, Das Greenhorn-Modul, VD Vol.V/4, Dez.89

Jörg Staben, Hilden

Markus Redeker aus Hamburg hat für das volksFORTH einen Z80-Assembler geschrieben und auch den 8080 volksFORTH Kern für den Z80 umgeschrieben. Der Assembler geht in seiner Grundkonzeption auf den Z280 Assembler von Heinz Schnitter zurück. Der Platzbedarf im Dictionary beträgt 3kB.

Ein Weg, wie man FORTH klein macht

--volksFORTH-83 lite--

Der folgende Quellcode für das volksFORTH-83 schränkt dessen Wörterbuch auf den Wortschatz der "FORTH-83 required words" ein. Dadurch ist es möglich Anfängern einen einfacheren Einstieg in das Programmieren mit Standard- FORTH-83 zu geben. Fortgeschrittene und Experten können damit Quellcode auf FORTH-83 Standard-Verträglichkeit prüfen.

Beim Vorstellen eines FORTH-Systems kommt es immer wieder zu folgender Situation: Nachdem ein paar erklärende Worte gefallen sind, soll nun ein Blick in das Wörterbuch des System geworfen werden. Das dazu nun eingegebene WORDS ist schier erschlagend: Mehrere Bildschirmseiten voll von Wortnamen rauschen über den Bildschirm. Das ist für den Anfänger und für den Experten wenig hilfreich.

Der Wunsch nach einem FORTH-System mit kleinem Wörterbuch kommt auf.

Der hier verfolgte Ansatz nimmt ein recht umfangreiches FORTH-System (volksFORTH-83) und schränkt nun konzeptuell dessen Wörterbuch ein. Zu diesem Zweck werden alle Worte, die in dem neuen FORTH-System definiert sein sollen ein weiteres Mal durch das Worte REDEFINE als ALIAS-Definitionen definiert. Werden die neuen Definitionen in neuen Vokabularen gemacht, und werden die alten Vokabulare später unwiederbringlich aus der Suchordnung verbannt, bleibt nur noch der Zugriff auf die redefinierten Worte -- das Wörterbuch ist konzeptuell kleiner geworden.

Um ein striktes FORTH-83-System zu erhalten bleibt dann nichts weiter zu tun, als alle Worte des "required wordsets" aufzuzählen [SCR 4,5]. Damit das noch etwas einfacher wird gibt es das Wort MEMBERS, das eine gegebene Anzahl von REDEFINEs macht

```
Scr 1   FORTH83.SCR
 0 \ volksFORTH-83 lite LOAD-Screen                    uh 31jan91
 1
 2 2 load
 3 3 5 thru
 4 6 load
 5
 6 Onlyforth  use forth.scr  forth root also forth definitions
 7
 8 | : hello
 9        full page  28 spaces ." volksFORTH-83 lite"
10        only forth also definitions decimal cr quit ;
11
12 ' hello is 'cold
13
14
15 1 scr !  r# off   savesystem forth83.com

 Scr 2   FORTH83.SCR
 0 \ redefine members                                  uh 31jan91
 1
 2
 3 | : of-what ( <name> -- <name> attr cfa )
 4        >in push  ' dup >name c@  swap ;
 5
 6 | : set-attribute ( attr -- )
 7        $C0 and   last @   under c@ or   swap c! ;
 8
 9 | : redefine ( <name> -- )   of-what Alias set-attribute ;
10
11 | : members ( n -- )  0 ?DO redefine LOOP ;
12
13
14
15

 Scr 3   FORTH83.SCR
 0 \ Redefine Root and Forth vocabularies   p.61ff  uh 31jan91
 1
 2 Vocabulary Forth   Forth also definitions
 3
 4 | Vocabulary Root    ' Root   Onlyforth   Alias Root
 5
 6 Forth also   Root also definitions
 7
 8 : only ( -- )  vp off   Root also ;
 9
10 3 members
11 forth also definitions
12
13
14
15
```

Ein Weg, wie man FORTH klein macht
–volksFORTH-83 lite–

```
 Scr 4   FORTH83.SCR
 0 \ Required Word Set    p.25ff                          uh 31jan91
 1
 2 Forth definitions
 3
 4 ( Nucleus Layer )  58 members
 5 ! * */ */mod + +! - / /mod 0< 0= 0> 1+ 1-
 6 2+ 2- 2/ < = > >r ?dup @ abs and c! c@ cmove
 7 cmove> count d+ d< depth dnegate drop dup execute exit
 8 fill i j max min mod negate not or over pick r>
 9 r@ roll rot swap u< um* um/mod xor
10
11 ( Device Layer ) 12 members
12 block buffer cr emit expect flush key save-buffers
13 space spaces type update
14
15

 Scr 5   FORTH83.SCR
 0 \ Required Word Set    p.25ff                          uh 31jan91
 1
 2 ( Interpreter Layer ) 32 members
 3 # #> #s #tib ' ( -trailing . .( <# >body >in
 4 abort base blk convert decimal definitions find forget
 5 forth forth-83 here hold load pad quit sign span tib u. word
 6
 7 ( Compiler Layer ) 30 members
 8 +LOOP , ." : ; abort" allot begin compile constant
 9 Create DO does> ELSE IF immediate leave Literal LOOP
10 REPEAT state THEN UNTIL Variable Vocabulary WHILE [ [']
11 [compile] ]
12
13
14
15

 Scr 6   FORTH83.SCR
 0 \ I can't live without that ...                        uh 31jan91
 1
 2 redefine words
 3 root definitions    redefine words   forth definitions
 4
 5 redefine savesystem
 6 redefine bye
 7 redefine list
 8 redefine .s
 9 redefine ed
10 redefine edit
11 \ redefine Onlyforth
12
13
14
15
```

[SCR 2]. Das ONLY-ALSO-Konzept ("exprimental proposal") [SCR 3] und einige nicht FORTH-83-Worte, die aber in ständiger Benutzung sind [SCR 6], werden ebenfalls noch definiert.

Der vorgestellte Quellcode läßt keine Lücke beim Schließen des Systems, so daß im neu auf dem Massenspeicher abgelegten FORTH-System (FORTH83.COM) keine Möglichkeit gibt, auf die alten Definitionen des Original-System zurückzugreifen. (Ein Schlupfloch kann leicht eingebaut werden, indem auch das Wort ONLY-FORTH redefiniert wird, das dann ja wieder die alten Vokabulare in die Suchordnung mit aufnimmt.)

Anfänger können sich auf diese Weise mit dem FORTH-83 Wortschatz, den man als den "Grund-Wortschatz FORTH" bezeichnen könnte, vertraut machen, ohne gleich von einem übergroßen System erschlagen zu werden. Dem Experten kommt das so entstandene FORTH-System ebenfalls zur Hilfe, will er Quellcode auf seine FORTH-83 Standard-Verträglichkeit prüfen.

Quellen:

[1] "ZAPPING THE F83 DICTIONARY", C.H.Ting, Offete Enterprises, FORML 1986 conference proceedings, Forth Interest Group, San Jose, 1986

[2] "FORTH-83 Standard", Forth Standards Team, Mountain View, 1983

Autor: Ulrich Hoffmann, Kiel

F-PC spricht deutsch

Dieses File enthält die Anpassungen von F-PC an die Eigenheiten der deutschen Sprache.

Folgende Änderungen werden durchgeführt:

1. Im Editor wird das Tastatur-Layout so geändert, daß die Umlaute direkt in den Text übernommen werden können. (Mit ALT-O A) erreicht man alle anderen Sonderzeichen.

2. Die Tabelle für die Großbuchstaben-Wandlung wird so angepaßt, daß auch die Umlaute richtig gewandelt werden.

3. Im Kommandozeilen-Editor wird das Tastatur-Layout ebenfalls angepaßt, damit auch dort die Umlaute eingebbar werden.

Ausstehende Änderungen:
Die Editor-Funktionen (ALT-O_L, ALT-O_U, ALT-O_W und ALT-O_C), die eine Groß/Klein-Wandlung im Text vornehmen, sind im File SEDCASE.SEQ definiert. Dort müßte eine Anpassung vorgenommen werden, um diese Funktionen auf Umlaute auszudehnen.

Autor: Ulrich Hoffmann , Kiel

```
decimal editor definitions  \ Umlaute äöüÄÖÜß im Editor erlauben.

\ Die überdeckten Funktionen könnten auf andere Tasten gelegt werden

: sed_itself       ( --- )  \ insert (even function) characters themselves
            keychar schr ;

132 fnset sed_itself   \ German Umlaut ä  \ überdeckt Ctrl PGUP  Scroll Up
148 fnset sed_itself   \ German Umlaut ö  \ überdeckt ALT T      Set TAB
129 fnset sed_itself   \ German Umlaut ü  \ ALT 0
142 fnset sed_itself   \ German Umlaut Ä  \
153 fnset sed_itself   \ German Umlaut Ö  \ überdeckt ALT P      Print Menu
154 fnset sed_itself   \ German Umlaut Ü  \
225 fnset sed_itself   \ German Umlaut ß  \ CTRL F4


forth definitions

\ Die Tabelle für Großwandlung für die Umlaute patchen

Ascii Ä   atbl Ascii ä +   c!
Ascii Ö  atbl Ascii ö +   c!
Ascii Ü   atbl Ascii ü +   c!
Ascii Ä   atbl Ascii Ä +   c!
Ascii Ö  atbl Ascii Ö +   c!
Ascii Ü  atbl Ascii Ü +   c!
Ascii ß   atbl Ascii ß +   c!


\ Umlaute im Zeileneditor erlauben

hidden also forth definitions

: ledit_itself ( --- ) \ insert (even function) characters themselves lchar har ;

>keys1

' ledit_itself  132 lkey!  \ ä
' ledit_itself  148 lkey!  \ ö
' ledit_itself  129 lkey!  \ ü
' ledit_itself  142 lkey!  \ Ä
' ledit_itself  153 lkey!  \ Ö

' ledit_itself  154 lkey!  \ Ü
' ledit_itself  225 lkey!  \ ß

>keys2

' ledit_itself  132 lkey!  \ ä
' ledit_itself  148 lkey!  \ ö
' ledit_itself  129 lkey!  \ ü
' ledit_itself  142 lkey!  \ Ä
' ledit_itself  153 lkey!  \ Ö
' ledit_itself  154 lkey!  \ Ü
' ledit_itself  225 lkey!  \ ß

only forth also definitions

.( Deutsche Anpassungen geladen! ) cr

mark empty
 \s  $Header:   D:/fpc/german.sev   1.0   25 Oct 1990 20:38:16  $
```

Ordentliche Mitgliederversammlung der Forth Gesellschaft e.V.

Wann:

Sonntag, den 14. April 1991 ab 10:00 Uhr

Wo:

Hotel Olympia
W-8046 Hochbrück bei München
Ingolstädter Landstraße 110

Telefon: 089/3 18 16-0

Tagesordnungspunkte

1. Rechenschaftsbericht des Direktoriums.
2. Kassenbericht und Bilanz 1990.
3. Aussprache und Entlastung des Direktoriums.
4. Wahl des Direktoriums.
5. DFÜ-Versorgung der Mitglieder mit Forth-Nachrichten.
6. Verschiedenes.

Das alljährliche Treffen beginnt schon am Freitag dem 12.April mit Vorträgen zu Forth aus Wissenschaft und Industrie.

Ab 12. April findet unser jährliches FORTH-Treffen statt. Die Tagung endet am 14. April mit der ordentlichen Mitgliederversammlung der FORTH-Gesellschaft e.V. Die Teilnahme an der Mitgliederversammlung ist kostenlos. Bitte kommen sie recht zahlreich und nehmen Sie sich auch Zeit für die Mitgliederversammlung, damit diese beschlußfähig ist. Es stehen wichtige Themen auf der Tagesordnung, die die Ziele der Gellschaft festlegen sollen.

Approximation durch Brüche

(orig. "Fractions")

Dieser Artikel beschreibt eine Methode, die es erlaubt einen Dezimalbruch der Form n,abcd... durch einen gewöhnlichen Bruch p/q (p,q einfach genaue Zahlen) bestmöglich anzunähern.

Stichworte
Dezimalbruch, Bruch, decimal fraction, fraction, volksFORTH-83

Ein Dezimalbruch n,abcd... soll durch einen gewöhnlich Bruch p/q (p,q einfach genaue Zahlen) ersetzt werden, sodaß beide Zahlen in ihrer Dezimaldarstellung eine gegebene Anzahl gemeinsamer Ziffern besitzen. Da dieses Problem keine oder aber auch mehrere Lösungen besitzen kann, wird eine Liste der möglichen Ergebnisse ausgegeben. Um die Auswahl des besten Ergebnisses zu vereinfachen, werden die beiden nächsten Dezimalstellen des Bruches ebenfalls angezeigt.

Der Dezimalbruch n,abcd... wird eingegeben durch n DECIMALS abcd... <Ret> (n < 25)

```
 Scr  1  FRACTION.SCR
 0 \ Fractions
 1
 2  Onlyforth
 3  1  3  +thru
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15

 Scr  2  FRACTION.SCR
 0 \ Fractions  input
 1
 2 : gcd ( u1  u2 -- gcd )
 3      BEGIN   ?dup   WHILE   under u/mod drop   REPEAT ;
 4 | Variable hit
 5 | : next-digit ( D N -- D R Q )
 6        base @ um*   2 pick   um/mod ;
 7 | : wrong-input ( f -- )
 8        0= abort"   - input: n decimals abcd.." ;
 9 | : input-number ( n -- )
10        depth wrong-input   dup &24 > abort" - max 24,..."
11        bl word count ?dup wrong-input   pad place
12        pad count bounds
13        ?DO   I c@ digit? wrong-input I c! LOOP
14        base @ *   pad 1+ dup c@   rot +   swap c! ;
15
```

Approximation durch Brüche
(orig. "Fractions")

Ein Beispiel:

Es wird ein Bruch gesucht, der die reelle Zahl e ersetzen soll (e=2,7182818284...). Durch die Eingabe von
2 DECIMALS 7182818 (28) wird (nach einiger Zeit) folgende Liste ausgegeben:

Man erkennt, daß 23225/8544 und 25946/9545 die Kandidaten für die beste Approximation von e sind

Zähler	Nenner	Ziffern
14962	5541	98
17783	6542	70
23225	8544	35
25946	9545	22
28667	10546	13
30504	7543	50

```
 Scr 6 FRACTION.SCR
 0 \\ Fractions, some examples
 1
 2    For pi = 3,14159265
 3    3 decimals 141592 ( 65)
 4    355    113 92    30508   9711 00    30863   9824 01   31218    993 02
 5 31573 10050 03    31928 10163 04    32283 10276 05   32638 10389 06
 6    355 113 9 uf.r   3,14159292
 7    355/113 approximates pi with an error 27 * 10^-8
 8
 9   For the periodic decimal fraction 0,435897435897...
10      0 decimals 435897435897
11        17    39 43
12        17 39 18 uf.r   0,435897435897435897
13        17/39 is exact
14
15


 Scr 7 FRACTION.SCR
 0 \\ Fractions
 1
 2 Abbreviations in the stack comments :
 3    N, D numerator and denominator; R, Q remainder and quotient
 4 GCD             Greatest common divisor
 5 HIT             flag
 6 NEXT-DIGIT   find next digit in N/D
 7                 base @ 2 pick */mod    is shorter, but slower
 8 WRONG-INPUT abort if the input is wrong
 9 INPUT-NUMBER   a number n,abcd... is input in the form
10               n INPUT-NUMBER abcd... <Ret>
11    The integer part, n should be less than 25
12    The decimal part is placed in PAD; ascii values converted to
13    digits and base * n added to the first digit
14
15
```

Approximation durch Brüche (orig. "Fractions")

(Die folgenden Dezimalstellen sind "35" bzw. "22", die von e "28"). Um nun die beste Approximation zu finden, berechnen wir beide Brüche jeweils auf zehn Dezimalstellen:

p q n UF.R

gibt einen positiven Bruch p/q mit n Dezimalstellen aus.

Nun ist ersichtlich, daß also 25946/9545 die beste Approximation von e ist.

23225 8544 10 UF.R 2,7182818352

25946 9545 10 UF.R 2,7182818229

In seinem Buch "Starting Forth" gibt Leo Brodie Approximationen für einige irrationale Zahlen an.
Ich habe das Wort DECIMALS benutzt um einige seiner Zahlen nachzurechnen:

```
 Scr 3 FRACTION.SCR
 0 \ Fractions output
 1
 2 | : .digit ( u -- )
 3         0 <# #s #> type ;
 4
 5 | : fraction-out ( R N D F -- N D )
 6       IF  2 spaces  over 6 u.r  dup 6 u.r  rot 2 spaces
 7           next-digit .digit  next-digit .digit  2 spaces
 8       ELSE  rot
 9       THEN  drop ;
10
11 | : found? ( N D R -- N D+1 ) \ ( N D -- N D )
12       hit @ IF  -rot 2dup gcd 1 = fraction-out  1+  THEN ;
13
14
15

 Scr 4 FRACTION.SCR
 0 \ Fractions user words
 1
 2 : decimals ( n -- )
 3     input-number cr  1 1
 4     BEGIN    over pad count bounds  hit on
 5       ?DO  next-digit
 6            dup I c@ > IF 2drop 1+              hit off  LEAVE THEN
 7                 I c@ < IF drop swap 1+ swap    hit off  LEAVE THEN
 8       LOOP
 9       found?   over 0<  over 0< or  stop? or
10     UNTIL  2drop ;
11
12 : uf.r ( p q n > )
13       >r  under u/mod .digit  ." ,"
14       r> 0 DO  next-digit .digit LOOP  2drop ;
15
```

Approximation durch Brüche (orig. "Fractions")

Zahl	Brodie	rel. Fehler * 10^-9	DECIMALS	rel. Fehler * 10^-9
pi = 3,14159...	355/113	85	unverändert	
sqrt 2 = 1,41421...	19601/13860	1,5	27720/19601	0,7
sqrt 3 = 1,73205...	18817/10864	1,1	unverändert	
sqrt 10= 3,16227...	22936/7253	5,7	31922/10063	2,8
e = 2,71828...	28667/10546	5,5	25946/9545	1,0

[1] "Starting Forth", Leo Brodie, Prentice Hall, New York 1981in deutsch
"Programmieren in Forth", Leo Brodie, Carl Hanser Verlag

Autor : Finn Berlev , DK - Farum / aus dem Englischen von Ulrich Hoffmann

```
Scr 8 FRACTION.SCR
 0 \\ Fractions
 1
 2 .DIGIT          output a positive number without trailing space
 3 FRACTION-OUT  output of the fraction with all decimals right
 4  if gcd = 1 :
 5  numerator, denominator followed by the two next digits
 6  If gcd > 1 : no output
 7
 8 FOUND?  If hit-flag set, use farction-out;
 9          increase denominator
10
11
12
13
14
15

Scr 9 FRACTION.SCR
 0 \\ Fractions
 1
 2 DECIMALS  ( n -- )   The number n,abcd... is input :
 3   input; numerator and denominator = 1
 4
 5
 6   If next digit is too big, then increase denominator
 7   If next digit is too small, then increase numerator
 8   Stop if numerator or denominator no longer is a single
 9   precision number.
10   Unsinged fractions is found, if 0< is replaced by 0=
11
12  UF.R  output a positive fraction p/q with n decimal digits
13
14
15
```

Fünf Beispiele zum Sortieren in Forth

Das Sortieren von Daten mag auf den ersten Blick als eine völlig unproblematische Sache erscheinen; in Wirklichkeit jedoch stellt effizientes Sortieren einer großen Menge von Daten ein Problem dar, für das es bis heute und wohl auch in Zukunft kein allgemein optimales Verfahren gibt. In den vergangenen Jahren wurde eine ganze Reihe von verschiedenen Sortieralgorithmen entwickelt, deren Leistungsfähigkeit deutliche Unterschiede aufweist. So gibt es Algorithmen, welche zwar beinahe sortierte Daten enorm schnell fertigsortieren können, sich aber als ausgesprochen langsam erweisen, wenn die Daten in völlig unsortierter Verteilung auftreten. Oder Methoden, deren Schnelligkeit erst dann zur Geltung kommt, wenn die Menge der zu sortierenden Daten sehr groß wird, während sie bei einer kleineren Datenmenge eher langsam arbeiten. Um dem Leser ein Gespür dafür zu geben, welche Methode für den jeweiligen Zweck die geeignetste ist, sollen hier die bekanntesten Sortierverfahren vorgestellt werden, zusammen mit einem kleinen Testprogramm, welches spezielle Eigenschaften der einzelnen Routinen deutlich macht.

Die verschiedenen Sortieralgorithmen lassen sich grundsätzlich in zwei verschiedene Leistungsklassen einteilen. Bei den sog. *direkten Methoden* handelt es sich um Algorithmen, die relativ einfach zu implementieren sind. Bekannte Vertreter hierfür sind der Bubblesort, der Shuttlesort, das Sortieren durch direktes Auswählen sowie Sortieren durch direktes Einfügen. Charakteristisch für diese Methoden ist, daß die zum Sortieren benötigte Zeit proportional zu

```
\ SORTTEST.SCR                                  HH 19:30 07.02.91

 Sortierverfahren für Zahlen einfacher Länge
 geschrieben im FORTH-83 Dialekt mit LMI PC/FORTH 3.2

 Befehle: ELEMENTE ( n -- )
          NORMAL ZUFALL INVERS  ( -- )
          INHALT                ( -- )
          ASORT BSORT ESORT SSORT QSORT  ( -- )
          ATEST BTEST ETEST STEST QTEST  ( -- )

\ Ladeblock                                     HH 19:30 07.02.91

  FORTH-83   : IT ;   2 15 THRU

  200 ELEMENTE          \ Jede Zahl von 2 bis 2000 ist erlaubt

\ Ausschließen von nicht mehr benötigten Headern

  EXCISE EXCHANGE EXCHANGE
  EXCISE SEED RND
  EXCISE Max#El End
  EXCISE Pos Pos
  EXCISE VALUE qsort
  EXCISE Stand .SEC
  EXCISE Sort TEST

\ Nützliche Definitionen                        HH 19:30 07.02.91

: EXCHANGE  ( i j -- )  2DUP  >R >R  >R @ R> @  R> ! R> ! ;
\ Vertauscht die in den Speicheradressen enthaltenen Zahlen


\\ Folgende Worte sind bereits im Wortschatz von PC/FORTH enthalten. Für Kompatibilität mit anderen Systemen werden sie
 hier noch einmal aufgeführt:

: NIP  ( n1 n2 -- n2 )  SWAP DROP ;
: PERFORM  ( adr -- )  @ EXECUTE ;
: RECURSE  ( -- )  LATEST NAME> , ; IMMEDIATE



\ Generator für Pseudo-Zufallszahlen            HH 19:30 07.02.91

  HEX

  VARIABLE SEED
\ Der Startwert dieser Variablen legt die gesamte Abfolge der
\ erscheinenden Zahlen fest. Es handelt sich also in Wirklich-
\ keit nur um Pseudo-Zufallszahlen.

: rnd  ( -- n )  SEED @  103 *  3 +  7FFF AND  DUP  SEED ! ;
\ Liefert Zufallszahlen im Bereich von 0 bis incl. 32767
```

Fünf Beispiele zum Sortieren in Forth

```
: RND  ( n -- n')  rnd M*  8000 UM/MOD  NIP ;
\ Liefert Zufallszahlen im Bereich von 0 bis excl. n'

  DECIMAL

\ Speicherplatz reservieren                    HH 19:30 07.02.91

  2000 CONSTANT Max#El  \ Obergrenze für Anzahl der Elemente

  HERE   Max#El 2* ALLOT   CONSTANT Anf
 Reserviert einen Speicherbereich und weist Anf per Stack die
\ Adresse der ersten zu sortierenden Zahl zu

  0 CONSTANT #El      \ Anzahl der zu sortierenden Elemente
  0 CONSTANT End      \ Adresse der letzten zu sortierenden Zahl

: ELEMENTE  ( n -- )    \ Versorgt #El und End mit neuem Inhalt
  2 MAX  Max#El MIN
  DUP              [ ' #El >BODY ] LITERAL  !
  1- 2*  Anf +   [ ' End >BODY ] LITERAL  ! ;

\ Inhalt des Arrays anzeigen                   HH 19:30 07.02.91

: INHALT  ( -- )
  CR
  0                                  \ Anfangswert für den Zähler
  End 2+  Anf
  DO  DUP  10 MOD  0=                \ Wieder 10 Zahlen ausgegeben?
      IF CR THEN                     \ Wenn ja, dann Zeilenvorschub
      1+                                 \ Zähler um 1 erhöhen
      I @  7 .R                      \ Adresseninhalt ausgeben
  2 +LOOP
  DROP                                   \ Zähler wegwerfen
  CR CR ;

\\ Mit diesem Befehl kann die Reihenfolge der Zahlen vor und
 nach einem Sortiervorgang eingesehen werden.

\ Werte in den Speicher schreiben              HH 19:30 07.02.91

: NORMAL  ( -- )
  0  End 2+ Anf DO  DUP I !  1+  2 +LOOP  DROP ;

: ZUFALL  ( -- )
  End 2+ Anf DO  #El RND  I !  2 +LOOP ;
: INVERS  ( -- )
  #El  End 2+ Anf DO  1-  DUP I !  2 +LOOP  DROP ;

\\ Mit diesen drei Befehlen können Zahlen in normaler, zufäl-
 liger und umgekehrter Reihenfolge in den Speicher geschrieben
 werden. Dies ermöglicht es, die für eine Sortiermethode spe-
 zifischen Eigenschaften herauszufinden.
```

n^2 ist,

wenn "n" die Anzahl der zu sortierenden Datenelemente darstellt. Ein Vergrößern der Datenmenge auf das zehnfache hat deshalb bereits hundertfache Sortierzeit zur Folge. Es ist klar, daß die direkten Methoden für das Sortieren größerer Datenmengen schnell uninteressant werden.
Bei *höheren Methoden* dagegen wächst die zum Sortieren benötigte Zeit langsamer an als quadratisch. Weil diese Algorithmen jedoch komplexer aufgebaut sind als die von direkten Methoden, kommt ihre Stärke erst dann voll zur Geltung, wenn auch eine entsprechend große Datenmenge vorliegt. Der Shellsort ist mit einer zu

$n^{1.2}$

proportionalen Sortierzeit ein bekanntes Beispiel für eine höhere Methode. Noch schneller arbeiten die beiden logarithmischen Methoden Heapsort und Quicksort. Ihre Sortierzeit ist proportional zu n*log(n), für sehr große Zahlen also praktisch proportional zur Anzahl der Daten. Von den höheren Methoden soll hier der Quicksort vorgestellt werden.

Die einzelnen Algorithmen

Hier folgt eine kurze Beschreibung der Algorithmen, auf denen die im Listing enthaltenen Routinen basieren. Ausführlich werden die Algorithmen in [1] besprochen. Zum besseren Verständnis der Routinen ist es hilfreich, kleine Papierstückchen mit Zahlen zu beschriften, und diese dann entsprechend den Algorithmen durchzusortieren.

ASORT: Sortieren durch direktes Auswählen

Hierbei wird das Array gedanklich in zwei Teile zerlegt. Der vordere Teil sei das bereits sortierte Zielarray, dahinter befindet sich das unsortierte Quellarray. Anfangs ist das Zielarray völlig

leer. Nun wird das gesamte Quellarray von links nach rechts durchlaufen und dabei die kleinste Zahl im Quellarray ermittelt. Diese Zahl wird mit der ersten Zahl des Quellarrays vertauscht, und anschließend die Grenze zwischen Ziel- und Quellarray um eine Position nach rechts versetzt. Der Vorgang wird so lange wiederholt, bis das Quellarray leer ist. Im Endeffekt muß also wiederholt das Quellarray durchsucht werden, um davon jeweils das korrekte Element auszuwählen und an eine bestimmte Position im Zielarray zu bringen.

ESORT:
Sortieren durch direktes Einfügen

Das Array wird auch hier in ein bereits sortiertes Zielarray, und ein rechts davon stehendes unsortiertes Quellarray zerlegt. Anfangs besteht das Quellarray gerade aus dem ersten Element des gesamten Arrays. Nun wird in jedem Lauf das erste Element des Quellarrays genommen und mit allen links von ihm stehenden Zahlen des Zielarrays verglichen. Ist eine dieser Zahlen kleiner als der Vergleichswert, so wurde die korrekte Einfügposition gefunden. Um Platz für die einzufügende Zahl zu schaffen, müssen einige Zahlen im Zielarray um eine Position nach rechts versetzt werden. Diese Verschiebungen können abwechseln mit den Vergleichen durchgeführt werden, wovon in der Routine ESORT Gebrauch gemacht wurde. Es erweist sich in Forth jedoch als günstiger, zunächst nur die Einfügposition zu suchen, und erst danach den gesamten angefallenen Zahlenstrang auf einmal zu versetzen. Für diesen Zweck kann dann nämlich der schnelle Befehl CMOVE eingesetzt werden. Die so programmierte Routine ETEST' arbeitet z.B. auf meinem Rechner 35 Prozent schneller als die normale Version. Sortieren durch direktes Einfügen arbeitet demnach gerade umgekehrt wie Sortieren durch direktes Auswählen: es wird bei jedem Lauf ein bestimmtes Element aus dem Quellar-

```
\ Sortieren durch Auswählen                    HH 19:30 07.02.91

\ N. Wirth: Algorithmen und Datenstrukturen
\ 4. Auflage 1986, S. 82

: ASORT  ( -- )
  End Anf
  DO  I DUP @                                          \ j x
      End 2+ I 2+
      DO  DUP I @  >
          IF  2DROP  I DUP @  THEN
      2 +LOOP                                          \ j'x'
      DROP                                             \ j'
      I EXCHANGE
  2 +LOOP ;


\ Bubblesort                                   HH 19:30 07.02.91

\ N. Wirth: Algorithmen und Datenstrukturen
\ 4. Auflage 1986, S. 84

: BSORT  ( -- )
  Anf 2+ End
  DO  I Anf
      DO  I @  I 2+ @  2DUP >
          IF  I !  I 2+ !  ELSE  2DROP  THEN
      2 +LOOP
  -2 +LOOP ;



\ Sortieren durch Einfügen                     HH 19:30 07.02.91

\ N. Wirth: Algorithmen und Datenstrukturen
\ 4. Auflage 1986, S. 79

  VARIABLE Pos                  \ Speicher für die Einfügposition

: ESORT  ( -- )
  End 2+ Anf 2+
  DO  Anf Pos !  I @
      Anf I 2-
      DO  DUP  I @  <
          IF  I @  I 2+ !  ELSE  I 2+ Pos !  LEAVE  THEN
      -2 +LOOP  Pos @  !
  2 +LOOP ;
```

Fünf Beispiele zum Sortieren in Forth

```
\ Schnellere Version von ESORT                    HH 19:30 07.02.91
\\
: ESORT' ( – )
   End 2+ Anf 2+
   DO   Anf Pos !  I @
        Anf I 2-
         DO   DUP   I @   <
              NOT IF   I 2+ Pos !   LEAVE   THEN
        -2 +LOOP   Pos @   DUP   DUP 2+   OVER I SWAP -   CMOVE>   !
   2 +LOOP ;

\\ Anstatt die Zahlen bereits während des Suchvorgangs einzeln
  nach rechts zu versetzen, wird hier erst einmal nur der Such-
  vorgang durchgeführt. Ist die Einschubstelle gefunden, wird
  mit Hilfe des schnellen Befehls CMOVE> die gesamte Zahlenkette
  auf einmal um eine Position nach rechts verschoben.



\ Shuttlesort                                     HH 19:30 07.02.91

\ Dr. Dobb's Toolbook of Forth I
\ 2. Auflage 1987, S. 42

: SSORT   ( adr1 adr2 – )
   End Anf
   DO   Anf I
        DO   I @   I 2+ @   2DUP >
             IF   I !   I 2+ !   ELSE   2DROP LEAVE   THEN
        -2 +LOOP
   2 +LOOP ;


\ Quicksort /1/                                   HH 19:30 07.02.91

\ N. Wirth: Algorithmen und Datenstrukturen
\ 4. Auflage 1986, S. 96

: VALUE   ( i j – x )              \ Vergleichswert in Arraymitte
   OVER -   2/ 2/ 2* +   @ ;

: INC_L   ( x i – x i' )           \ Inkrementiert linke Grenze
   BEGIN   2DUP @ >   WHILE   2+   REPEAT ;

: DEC_R   ( x j – x j' )           \ Dekrementiert rechte Grenze
   BEGIN   2DUP @ <   WHILE   2-   REPEAT ;

: ARRANGE   ( i'RLj' -- i'RLj' | Lj'i'R )   \ Kürzeren Ast zuerst
   2OVER 2OVER   - +   U> IF 2SWAP THEN ;
```

ray entnommen, die Einfügposition steht jedoch noch nicht fest. Um diese zu ermitteln, muß jetzt das Zielarray durchsucht werden.

BSORT:
Bubblesort
Bubblesort ist eine sehr einfachste Routine zum Sortieren von Daten. Hier wird das Array von links nach rechts durchlaufen. Immer wenn ein Paar von Zahlen nicht in korrekter Reihenfolge nebeneinander steht, werden diese zwei Zahlen miteinander vertauscht. Dadurch wandert im ersten Durchlauf die größte Zahl ganz an das rechte Ende des Arrays, im nächsten die zweitgrößte Zahl an die zweitletzte Position, etc. Die einzelnen Läufe starten immer ganz von links, das rechte Ende wird dagegen nach jedem Lauf um eine Position nach links versetzt, so lange, bis alle Zahlen sortiert sind.

SSORT:
Shuttlesort
Eine Verbesserung von Bubblesort stellt der Shuttlesort dar. Auch hier wird das Array von links nach rechts durchlaufen, wobei wieder Paare nebeneinander stehender Zahlen betrachtet werden. Ist es hier jedoch einmal nötig, ein Zahlenpaar zu vertauschen, dann wird die Richtung gewechselt und so lange vertauschend nach links gelaufen, bis die betrachtete Zahl an seiner korrekten Position angekommen ist. Danach geht es wieder von der Stelle aus nach rechts weiter, wo vorher abgebrochen werden mußte. Wenn man am rechten Ende angekommt, sind alle Zahlen sortiert.

QSORT:
Quicksort
Quicksort ist die einzige hier vorgestellte höhere Sortiermethode und deutlich komplexer als die anderen

hier vorgestellten Methoden. Man startet mit einem Array, das die Grenzen L und R habe. Nun wird mittels VALUE zunächst ein Vergleichswert x aus der Mitte des Arrays herausgegriffen. Das Array wird anschließend mit INC_L so lange von links aus durchsucht, bis ein Element

$$x_i \geq x$$

gefunden wird, und mit DEC_R so lange von rechts, bis ein Element

$$x_j$$

gefunden wird. Diese beiden Elemente werden nun durch die Routine EXCHANGE miteinander vertauscht. Das Suchen und Vertauschen wird so lange wiederholt, bis sich die beiden Zeiger i' und j' irgendwo im Array treffen. Im Endeffekt wurde das Array dadurch in zwei Teile zerlegt: im linken Teil L bis j' sind nun alle Zahlen kleiner als x, im rechten Teil i' bis R sind sie alle größer als x. Jetzt braucht sich Quicksort mit diesen neuen Grenzen nur noch selbst rekursiv aufrufen, um auch die beiden Teilarrays weiter zu sortieren. Die Rekursionen werden so lange durchgeführt, bis die Teilarrays lediglich aus einem einzigen Element bestehen. Dann befinden sich alle Zahlen in korrekter Reihenfolge.
Vor dem rekursiven Aufruf von Quicksort empfiehlt es sich, die auf dem Stack liegenden neuen Grenzen mittels ARRANGE so umzuordnen, daß zuerst das kleinere Teilarray fertigsortiert wird. Dies verhindert ein übermäßiges Anwachsen von Daten auf dem Parameterstack und damit einen eventuellen Stacküberlauf. Weil Quicksort bevorzugt zum Sortieren großer Datenmengen eingesetzt wird, sollte auf diese Schutzmaßnahme nicht verzichtet werden.

Bemerkungen zum Listing
Zunächst ist es wichtig zu wissen, daß alle hier vorgestellten Routinen eine Zahlenlänge von 16 Bit voraussetzen. Sollte Ihr System mit einer Zahlenlän-

```
\ Quicksort /2/                                    HH 19:30 07.02.91

: qsort  ( L R -- )
  2DUP >R >R  2DUP VALUE   R> R>                          \ L R x L R
  BEGIN  >R  INC_L  R> SWAP >R  DEC_R  R> SWAP    \ L R x i j
         2DUP U> NOT
         IF  2DUP EXCHANGE 2- SWAP 2+ SWAP  THEN
         2DUP U>
  UNTIL                                                  \ L R x i'j'
  >R >R DROP R> SWAP ROT R>  ARRANGE
  2DUP U< IF RECURSE ELSE 2DROP THEN
  2DUP U< IF RECURSE ELSE 2DROP THEN ;

: QSORT  ( -- )  Anf End qsort ;


\ Worte für die Zeitmessung                    HH 19:30 07.02.91
\ Dieser Block ist nur für PC/FORTH geeignet. Bei anderen Sy-
\ stemen müssen geeignete Ersatzdefinitionen verwendet werden.
  2VARIABLE Stand

: csec  ( n1 n2 -- d )
  SPLIT 100 * + S>D   ROT SPLIT 60 * + 6000 M*   D+ ;

: INIT  ( -- )  @TIME  Stand 2! ;
\ Setzt den Stand der Stoppuhr auf Null zurück

: GONE  ( -- d )  @TIME csec  Stand 2@ csec  D- ;
\ Legt die vergangene Zeit in Zentisekunden auf den Stack

: .SEC  ( -- )  <« » « ASCII . HOLD »S #> TYPE SPACE ." sec" ;
 Gibt die auf dem Stack liegende Zeit formatiert aus


\ Sortiertests mit Zeitmessung                 HH 19:30 07.02.91

  VARIABLE Sort                \ Speichert CFA der Sortiermethode
: SORT  ( -- )  Sort PERFORM ;          \ Vektorielle Ausführung

: TEST  ( -- )
  CR ." Normal: " NORMAL INIT SORT GONE .SEC
  CR ." Zufall: " 3 0 DO  ZUFALL INIT SORT GONE  LOOP
                  D+ D+  3 D/  .SEC
  CR ." Invers: " INVERS INIT SORT GONE .SEC CR ;

: ATEST  ( -- )  ['] ASORT Sort !  TEST ;
: BTEST  ( -- )  ['] BSORT Sort !  TEST ;
: ETEST  ( -- )  ['] ESORT Sort !  TEST ;
: STEST  ( -- )  ['] SSORT Sort !  TEST ;
: QTEST  ( -- )  ['] QSORT Sort !  TEST ;
```

ge von 32 Bit arbeiten, so müssen Sie einige Modifikationen durchführen. Im wesentlichen bestehen diese darin, Ausdrücke wie 2+, 2-, 2*, 2 +LOOP etc. durch verdoppelte Werte zu ersetzen. Ein zusätzliches Problem tritt beim Quicksort auf. Die Phrase 2/ 2* in dem Wort VALUE garantiert zwar, daß eine zuvor berechnete Adresse danach auch wirklich auf den Anfang einer 16 Bit Zahl zeigt. Bei längeren Zahlen funktioniert dieser Trick jedoch nicht, und es ist einfacher, hier über ein herkömmlich indiziertes Array auf die zu sortierenden Daten zuzugreifen.

In dem hier betrachteten Spezialfall des Sortierens von 16 Bit langen Zahlen belegen die Daten auf dem Stack gerade gleich viel Platz wie dort ebenfalls abgelegte Adressen. Deshalb könnten gemischte Zahlenpaare ohne weiteres mit Operatoren wie SWAP, ROT, OVER etc. bearbeitet werden. Eine Routine zum Vertauschen von zwei in den Adressen

i und j

gespeicherten Zahlen könnte z.B. wie folgt definiert werden:

```
EXCHANGE ( i j -- )
OVER @  OVER @
-ROT
SWAP !  SWAP ! ;
```

Derartige Praktiken wurden jedoch nach Möglichkeit vermieden. Stattdessen wurde in diesem und analogen Fällen bevorzugt mit dem Returnstack gearbeitet:

```
: EXCHANGE ( i j -- )
2DUP  R  R
R @ R @
R ! R ! ;
```

Die Anzahl der zu sortierenden Elemente und die sich daraus ergebende Adresse des letzten Elements wird in den Konstanten #El bzw. End gespeichert. Zur Speicherung von veränderlichen Werten werden zwar normalerweise Variablen eingesetzt, durch die Verwendung von Konstanten lassen sich die Routinen jedoch übersichtlicher gestalten. Das Wort ELEMENTE zeigt dann auch, wie der Wert dieser 'Konstanten' dennoch geändert werden kann. In PC/FORTH ginge dies mit der Anweisung EQU sogar noch bequemer. Der Ausdruck '400 ELEMENTE' z.B. sorgt dafür, daß die Routinen jeweils 400 Zahlen sortieren. Die Worte NORMAL, ZUFALL und INVERS füllen die Arrays mit Zahlen in unterschiedlicher Anordnung. Damit der Inhalt des Arrays vor und nach einem Sortiervorgang kontrolliert werden kann, wurde eine Routine namens INHALT definiert.

Alle Definitionen bis einschließlich QSORT verwenden praktisch ausschließlich Standardworte und sollten auf jedem 16 Bit System laufen. Die zur Zeitmessung eingesetzten Worte INIT, GONE und .SEC dagegen arbeiten in der abgedruckten Form nur mit PC/FORTH von LMI. Besitzen Sie ein anderes System, so müssen Sie geeignete Ersatzworte dafür finden. Das auf diese drei Worte aufbauende Benchmarkprogramm TEST ist übrigens ein nettes Beispiel für das *vektorielle* Ausführen einer Routine. Anstatt für jedes betrachtete Sortierverfahren ein eigenes Testprogramm in der Art von TEST zu schreiben, wurde nur diese eine Routine definiert, jedoch für ein noch unbestimmtes Sortierverfahren namens SORT. Welchen Sortiervorgang diese Routine wirklich repräsentieren soll wird erst unmittelbar vor dem Aufruf von TEST festgelegt. Weil die für das Sortieren zufällig verteilter Zahlen benötigt Zeit etwas streut, werden in diesem Fall jeweils drei Läufe durchgeführt und anschließend der Mittelwert daraus gebildet.

Leistungsvergleich

Im folgenden sind die Ergebnisse der Benchmark-Tests aufgeführt, ermittelt durch einen mit 12 MHz getakteten PC/AT kompatiblen Computer. Die Zahlenwerte beschreiben die zum Sortieren von anfangs in normaler, zufälliger beziehungsweise inverser Reihenfolge vorliegenden Zahlen benötigte Zeit in Sekunden.

200 ELEMENTE
ASORT: 0.44 / 0.46 / 0.55
BSORT: 0.69 / 0.82 / 0.98
ESORT: 0.02 / 0.38 / 0.77
SSORT: 0.01 / 0.49 / 0.97
QSORT: 0.05 / 0.10 / 0.07

2000 ELEMENTE
ASORT: 43.39 / 43.50 / 54.32
BSORT: 69.09 / 82.86 / 96.61
ESORT: 0.16 / 37.84 / 75.58
SSORT: 0.11 / 49.16 / 96.67
QSORT: 0.60 / 1.31 / 0.71

Das Sortieren durch *direktes* Auswählen arbeitet demnach zufriedenstellend für Zahlen in zufälliger oder inverser Anordnung. Aus einer Vorsortierung kann kein Nutzen gezogen werden. In derartigen Fällen sollte dieses Verfahren nicht eingesetzt werden.

Sortieren durch *direktes Einfügen* dagegen, bevorzugt sortierte oder fast sortierte Zahlen. Es bearbeitet auch zufällig angeordnete Zahlen schnell, sollte aber vermieden werden, wenn die zu sortierenden Zahlen bevorzugt in inverser Anordnung auftreten.

Der populäre *Bubblesort* erweist sich als die langsamste aller hier vorgestellten Sortiermethoden. Das schlechte Verhalten liegt zum einen darin begründet, daß die häufigen Zahlenvertauschungen viel Zeit in Anspruch nehmen. Die andere Schwäche von Bubblesort ist aber auch, daß ein bereits sortiertes Array nicht als solches erkannt wird; hier werden trotzdem alle Läufe durchgeführt.

Die verbesserte Version *Shuttlesort* dagegen erkennt ein sortiertes Array auf Anhieb und führt in diesem Fall keine einzige unnötige Operation durch. Der Shuttlesort ist die schnellste Methode überhaupt, um sortierte oder beinahe sortierte Zahlen zu be-

Fünf Beispiele für Sortieren in FORTH

arbeiten. Auch bei zufälliger Anordnung arbeitet er noch zufriedenstellend, dagegen ist seine Leistung bei invers angeordneten Zahlen genauso schlecht wie die von Bubblesort.

Alle vier genannten Sortierverfahren sind wegen ihrer zu

$$n^2$$

proportionalen Sortierzeit nicht für größere Datenmengen geeignet. Der Einsatz von *Quicksort* dagegen lohnt sich erst in diesem Fall, wo er dann auch außerordentlich schnell arbeitet. Auffällig ist übrigens die Tatsache, daß er zwar invers angeordnete Zahlen fast genauso schnell sortiert wie normale, für Zahlen in zufälliger Anordnung jedoch mehr Zeit benötigt.

Quellen

[1] Niklaus Wirth: Algorithmen und Datenstrukturen, 4. Auflage 1986 Teubner, Stuttgart

[2] Mark I. Manning: 'THE FORTH SORT' Dr. Dobb's Toolbook of Forth ISecond Edition 1987 M&T Publishing, California

[3] Wil Baden: 'Quicksort and Swords' Dr. Dobb's Toolbook of Forth II First Edition 1987 M&T Publishing, California

[4] Donald Kanth 'The Art of Compute Programming, III Sorting and Searching, Addison & Wesley, Menlo Parc, 1973.

Autor: Heinrich Hohl, Konstanz

Gewinner des Harris Design-Wettbewerb

RTX2001A

Der von 'Harris Semiconductor' zusammen mit 'elektronik industrie' durchgeführte Design-Wettbewerb rund um den RTX2001A Mikrocontroller ist abgeschlossen. Eine Jury besetzt aus Mitarbeitern von 'Harris', der Redaktion 'elektronik industrie'und der Fachhochschule Regensburg hat die Gewinner ermittelt.

Kategorie Software:

1. Preis Herr Dr. J. Friedrich und Herr Eckhardt, 7054 Korb
   für ihre Applikation und Muster-Lage-Erkennungssystem.

2. Preis Firma N. Leber, 8000 München 2
   für die Applikation Meßwertvorhersage zur Optimierung technischer Prozesse.

3. Preis Ing.-Büro B. Pallaske, 1000 Berlin 1
   für die Applikation Ultraschallraumabtaster.

Kategorie Hardware:

1. Preis Herr H. Schlote-Meißner, 3000 Hannover
   für die Applikation Aufbau einer seismischen Meßstation.

2. Preis Herr Dr. M. Ohsmann, 5100 Aachen
   für die Applikation Aufbau eines RDS-Signalgenerators plus Coder.

3. Preis Herr A. Zilker, 8044 Unterschleißheim
   für seine Applikation Drei-Achs-Schrittmotor-Controller.

Alle Preisträger erhalten für 1991 freie Mitgliedschaft in der FORTH-Gesellschaft e.V.

Herzlichen Glückwunsch auch von der FORTH-Gesellschaft e.V.

Ulrike Schnitter
-FORTH-Büro-

Universelles Filterwerkzeug

Stichworte: Textfilter, FFORTH

Einstieg:

-0.000258,-0.000268,-0.000224,
-0.000176,-0.000162,-0.000164,
-0.000156,-0.000154,-0.000156,
-0.000118,-0.00012,-0.0001,
-0.000104,-0.000128,-0.00014,
-0.000156,-0.000168,-0.00016,
-0.000142,-0.000114,-0.000096,
-0.000116

Es gibt wirklich Leute, die haben solche Probleme. Und im Gegensatz zu "Listenbehandlern" [1] brauchen diese Leute auch tatsächlich eine Lösung, denn die gezeigten Zahlen kommen in purem ASCII-Format aus irgendeiner Maschine und sollen nun in ein Array. Und das auch noch schnell. Natürlich wollen wir hier kein Spezialwerkzeug zur "ASCII-Zahlenfile-in-ein-Feld-packen-Problematik" präsentieren, sondern ein möglichst breit verwendbares Hilfsmittel mit hohem Nutzwert. Dazu muß zunächst das Problem allgemeiner definiert werden:

Daten einer sequentiellen Datenquelle sollen gefiltert werden. Um das Werkzeug auf das eigene System anzupassen, braucht man einige "Vorß"-kenntnisse:

- Man muß wissen, wie man in seinem System Dateien öffnet, aus ihnen liest und sie wieder schließt.
- Man muß wissen, wie das eigene FORTH-System die Compilerstrukturen wie DO...LOOP, IF...THEN oder : und ; absichert. Da gibt es im wesentlichen zwei Varianten: Einmal mit CSP und das andere Mal mit ?PAIRS, so wie es im hier benutzten FFORTH verwendet wird.
- Man muß wissen, wie das ';' im eigenen FORTH-System definiert ist oder es sich anschauen können. Das ';' wird zum Abschreiben oder zum Patchen gebraucht.
- Man muß eine Vorstellung davon haben, was und warum in einem FORTH-System auf dem Return-Stack passiert.

Testfrage:
Weshalb wird RDROP nicht so definiert?

```
: RDROP  r> drop ;   \ Falsch!
```

Prinzip:
Kernstück des Verfahrens ist eine Tabelle, die für jedes mögliche Eingangszeichen eine bearbeitende Prozedur enthält. Die aus dem Eingabestrom kommenden Zeichen geben an, welche Prozedur auszuführen ist. (Im einfachsten Fall sind alle Prozeduren gleich und tun gar nichts. Dies ist dann die ideale Datensenke). Die 'Intelligenz' des Filters steckt natürlich in diesen Prozeduren, die für jede Anwendung zu erstellen sind. Hierbei kann einem niemand helfen (siehe Beispiele im Listing). Man kann es aber leicht oder schwer haben. Höchste Leichtigkeit soll die im Folgenden entwickelte Lösung zeigen.

Lösungsansatz:
In einem ersten Ansatz würde man in einer Schleife bis zum Dateiende Zeichen für Zeichen lesen und mit diesen als Index durch die Tabelle springen:

```
BEGIN
  ReadChar dup ?eof not
  \ Zeichen für Zeichen lesen
WHILE
  cells FilterTable +
  \ Adresse berechnen
   Perform
   \ Tabellenprozedur ausführen
REPEAT drop
```

So möchten wir das nicht machen.Wir möchten uns nämlich die Schleife und die Rückkehr aus der Tabellenprozedur ersparen, da dies, zumindest im akademischen Sinne, zu einer Beschleunigung führt (weniger machen -> schneller fertig sein). Obige Lösung würde man auch in jeder anderen Sprache versuchen, obwohl man dabei bereits auf Schwierigkeiten bei der Parameterübergabe zwischen den Tabellenprozeduren stoßen kann. Dieser Weg ist also für einen FORTH-Programmierer völlig reizlos. Die höher gesteckten Ziele erreicht man, indem man die Tabellenprozeduren sowohl das Zeichenlesen als auch den Sprung durch die Tabelle selbst durchführen läßt. Dann muß man nur noch die erste Tabellenprozedur anstoßen und der Rest führt sich sozusagen selbst aus. Eine Tabellenprozedur könnte also so aussehen (z.B. die für das Zeichen 'e')

```
: handle-'e' ( -- )
     Verarbeite-'e'
\ kein Kommentar
 ReadChar dup ?eof
 \ Zeichen lesen
 IF exit THEN
 \ bei EOF zurück
 cells FilterTable +
 \ Adresse berechnen ...
      Perform ;
 \ ... und ausführen
```

So wollen wir es auch nicht machen, da sich bei dieser Lösung der Returnstack aufbaut, solange Zeichen gelesen werden, denn PERFORM hinterlegt bekanntlich die Rücksprungadresse auf dem Returnstack. Beim Erreichen des Dateiendes, wenn es überhaupt vor dem Absturz noch so weit kommt, führt dies zu einer Rücksprungkaskade, die wir ja gerade vermeiden wollten.Um dies zu verhindern, führen wir den Prozeduraufruf nicht selbst aus, sondern lassen das vom ';' erledigen. Dazu teilen wir ';' die

Universelles Filterwerkzeug

gewünschte Sprungadresse auf dem Returnstack mit:

```
: handle-'e' ( -- )
      Verarbeite-'e'
\ kein Kommentar
     ReadChar dup ?eof
\ Zeichen lesen
     IF exit THEN
\ bei EOF zurück
     cells FilterTable +
\ Adresse berechnen ...
     @ >R ;
\ ... und hinspringen.
```

So wollen wir das aber auch nicht machen. Viel schöner ist es doch so, wie wir es endlich gemacht haben:

```
: handle-'e'
     Verarbeite-'e' ;>
```

';>' spricht man wohl am besten 'exit to next'. Alle Tabellenprozeduren werden einfach mit ';>' anstelle von ';' beendet. Damit die Performance stimmt, lassen wir das von ';>' kompilierrte '(;>)' zusätzlich die Eingabedatei über einen Puffer ziehen, da zeichenweises Lesen ein Graus für jedes Betriebsystem ist. Der Puffer kann blockweise gefüllt werden.

Hilfsmittel:

Prozedurtabellen müssen natürlich auch erstellt und bearbeitet werden. Damit dies nicht in mühsame Handarbeit ausartet, nehmen wir FORTH-typische Hilfsmittel. Weil diese Tabellen oft für die meisten Zeichen dieselbe Prozedur enthalten, bietet es sich an, einen initialisierenden Filtertabellenersteller zu definieren:'

<Prozedur> Filter: <Filtername>

erzeugt eine Prozedurtabelle, die an allen Stellen mit der angegebenen Prozedur besetzt ist. Die Ausführung von <Filtername> macht diese Tabelle auch direkt zur aktuellen Tabelle.Um einzelne Prozeduren oder ganze Bereiche von Prozeduren zu ändern, benutzen wir die Worte

<index> runs <Prozedur>
<from.index> ... <to.index>
alltogether <Prozedur>

Kurz noch zum Abschluß: In C stößt man auf große Schwierigkeiten, wenn man dieses Verfahren anwenden will, da sich der (Return-)Stack da nicht so einfach explizit manipulieren läßt. Die Nützlichkeit des Filterutilities geht so weit, daß der Autor es sogar schon selbst benutzt hat, um 'mal eben' einen Pretty-Printer für ins GEMDOS exportierte F68K-Blocks zu realisieren. Die eingangs geschilderte "ASCII-Zahlenfile-in-ein-Feld-packen-Problematik" ist sogar authentisch und der Probleminhaber mit der Lösung (Beispiel 3) voll zufrieden.

Quellen:

[1] "Linked Actions", J. Plewe, F. Stüss, J. Staben, VD VI/3,VI/4, Forth Gesellschaft eV, München 1990

Autoren: J² Plewe/Staben

Die unendliche Geschichte oder Lokale Variablen

Teil MCXVII

Stichworte: lokale Variablen, F68K

ANS-FORTH wird lokale Variablen vorschreiben (laut mündlicher Mitteilung von Martin Tracy, FORTH-Tagung '90). Kein Systemanbieter wird auf Dauer darum herumkommen, dieses nützliche Feature zu implementieren. Eine Menge ist in den letzten VD's zu diesem Thema veröffentlicht worden [2,3,4]. Auch in einem früheren Heft [1] konnte man schon eine elegante Implementation des dynamischen Duos Hoffman/Schleisiek-Kern bewundern und abtippen. Aufgrund dieser Veröffentlichungsdichte möchte ich auf die Diskussion lokaler Parameter/Variablen hier verzichten. Im folgenden heißt einfach alles 'lokale Variable', ob sie nun variabel ist oder nicht.

Alle diese Lösungen haben eines gemeinsam: sie entsprechen syntaktisch nicht dem kommenden ANS-Standard (so wie ich ihn verstanden habe, s.o.):

```
: ADD ( a b -- a+b )
   LOCAL B  LOCAL A
   A B + ;
```

Das definierende Wort LOCAL ließ sich nun auf dem M68000-Sharewaresystem F68K überraschend kurz implementieren (hier in einer READ-ONLY-Variante). Dabei wird zunächst einmal mit dem Wort 'HEADER:', das auch in 'CREATE' verwendet wird, ein Header für die lokale Variable erzeugt. In F68K kommt dabei der Umstand, daß hier Code und Daten getrennt sind, erleichternd zum Tragen, da auf die Headererzeugung in einem Heap a la volksForth verzichtet werden kann. Ein Header kann in F68K die Codeerzeugung nicht stören. Der Header wird einfach erzeugt wie immer - fertig!

Wohlgemerkt, dies ist immer noch die Definition der lokalen Variablen und noch nicht die Referenz!

Nachdem dieser Code erzeugt wurde, wird die Adresse des 'Call RT_LOC@' (oben mit '*' markiert) in die CFA des soeben erzeugten Headers eingetragen. Damit ist die Definition der lokalen Variablen abgeschlossen.

Wird nun die lokale Variable im Fortgang der Definition referenziert ('A B +') so findet der Compiler den Header der Variablen (z.B. 'A') und kompiliert einen Sprung zu der Adresse, die die CFA des Headers angibt (native code!). Das ist alles!

Fast nebenbei hat das erste LOCAL zu Beginn den aktuellen DP und die Adresse des zuletzt definierten Wortes gesichert, die nach der Ausführung von ';' wieder restauriert werden. Damit sind auch die Header der lokalen Variablen wieder verschwunden. Leider mußte dafür ';' redefiniert werden.

Zur späteren Laufzeit (z.B. '3 5 ADD') wird zunächst RT_INIT ausgeführt.

RT_INIT
hat folgende einfache Aufgabe:

- 4 Byte auf dem Returnstack reservieren und mit dem Wert auf dem Datenstack initialisieren
- die Adresse dieser Bytes als In-line Adresse für RT_LOC@ patchen

Nun wird folgender Code für die lokale Variable erzeugt

```
+--------------+--------------+----------------+
| Call RT_INIT | Call RT_LOC@ | In-line Adresse |
+--------------+--------------+----------------+
                      ^
                      |
                      *
```

- die Einsprungadresse einer Aufräumroutine (FREELOCAL) auf dem Returnstack ablegen
- den Rücksprung-IP hinter die In-line Adresse zu setzen. Damit wird RT_LOC@ übersprungen. RT_LOC@ dient zur späteren Referenz und soll während der Initialisierung noch nicht ausgeführt werden.

Damit ist die lokale Variable auf dem Returnstack installiert und mit dem obersten Wert auf dem Datenstack initialisiert. Das Wort wird nun mit der nächsten Anweisung hinter der LOCAL-Definition fortgesetzt. Bei der Referenz der lokalen Variablen springt die Maschine zu dem Aufruf von RT_LOC@

(sprich: 'RunTime_LOCal-fetch')

im Initialisierungsteil der Variablen. Dessen Aufgabe ist wiederum einfach:

- hole die In-line Adresse (von RT_INIT gepatcht)
- hole den Wert der dort steht (von RT_INIT dort hinterlegt).

Damit liegt der Wert der lokalen Variablen auf dem Stack. Beim Erreichen des ';' liegen nun noch von jeder lokalen Variablen die Adresse eines FREELOCAL auf dem Returnstack (von RT_INIT dort hinterlegt), die damit ausgeführt werden, um für die lebenswichtige Ordnung auf dem Returnstack zu sorgen.Auch wenn diese Beschreibung wieder einmal verwirrt, ist das Prinzip ganz einfach. Leider

Stellengesuch:

Beruf:
Diplom Physiker, 26 Jahre, verh.

Sprachkenntnisse:
Englisch und Französisch

Kommunikation:
ASSEMBLER, C, LISP, PASCAL, OCCAM, FORTH

Spezialisiert auf:
Echtzeitprogrammierung, Parallelrechner und Medizintechnik

Hobby :
FORTH

Motiviert und flexibel:
Jörg Plewe
Tel.: 02 08-42 35 14

Die unendliche Geschichte oder Lokale Variablen

sind bei der Kodierung allerlei Returnstackmanipulationen und für F68K zusätzlich Adressumrechnungen notwendig gewesen, so daß auch das Listing auf den ersten Blick nicht den Durchblick gewährt. Ein zweiter Blick kann aber schon viel weiter helfen.

Bewertung:
Vorteile und Nachteile.

Zuerst die Vorteile:

- das Prinzip ist einfach
- 'sauberer' Ansatz ohne Steuer
- oder Kontrollvariablen zur Laufzeit
- die Implementation ist kurz (ein Abend)
- mit einfachen Mitteln kann die Methode erweitert werden. Ein paar Zeilen mehr und es sind auch Schreibzugriffe (via 'TO', wodurch die Variable erst variabel wird) oder andere Typen möglich (im Listing für FLOAT schon drin)
- Implementation wird für 32Bit-Integers mit Schreibzugriffen laufzeitoptimiert in den F68K-Kern übernommen
- leichte Portierbarkeit auch auf nicht-native-code-Systeme
- keine Einschränkung durch returnstack-ändernde Worte wie DO/LOOP
- hat Spaß gemacht.

Nachteile:

- hoher Speicherbedarf: in F68K für jede lokale Variable etwa 16 Bytes Laufzeitcode; reduziert sich drastisch bei gefädelten 16Bit-Systemen (6 Bytes)
- hoher Laufzeitbedarf: jede Variable wird einzeln allociert, initialisiert und wieder disallociert. Dies liegt an der Syntax, die kein einleitendes und abschlieendes Zeichen für die Definition der lokalen Variablen "en bloc" zuläßt, wie dies z.B. in [1,3,4] möglich war. Jede Referenz braucht zudem mindestens zwei CALLs: CALL Referenz - CALL RT_LOC@. Lokale Variablen ersparen aber in der Regel auch einige 'Swoperatoren', so daß sich der erhöhte Laufzeitbedarf relativiert.
- bin spät ins Bett gekommen

Quellen:

[1] U. Hoffman, K. Schleisiek-Kern "Parameter und lokale Variable in Forth", Vierte Dimension, Volume IV, Nr.1, März '88

[2] J. Yli-Nokari, "Lokale Variablen und Argumente", Vierte Dimension, Volume VI, Nr. 2, Juni '90

[3] J. R. Hayes "Lokale Variablen - ein anderes Verfahren", Vierte Dimension, Volume VI, Nr. 2, Juni '90

[4] R. Lehnhardt "Lokale Variablen" Vierte Dimension, Volume VI, Nr. 4, Dezember '90

Autor: Jörg Plewe

Jetzt verfügbar!

Super 8 ™-FORTH in Silicon

- ***FORTH*** *im 8Kbyte On-chip ROM*
- *Extern wird nur ein RAM, ein V.24-Treiber und ein 19,6608 MHz Quarz benötigt*
- *3 Ports verfügbar*
- *Transparente Entwicklungsumgebung kompiliert ROM-fähigen Code*
- *Zugriff auf alle On-chip Resourcen*
- *ausführliches Handbuch*
- *Quellcode von S8-Assembler und S8-FORTH werden mitgeliefert*
- *Muster 0887520PSC erhältlich nur bei:*

Klaus Kohl

Tel.: 0 82 33-3 05 24

oder

Ulrike Schnitter

Tel.: 0 89-3 10 33 85

the almighty FORTH:

für alle Atari ST/TT

Die Entwicklung von bigFORTH bleibt nicht stehen. Wenn Sie diese Zeilen lesen, wird bigFORTH Version 1.10 bereits erhältlich sein. Neben den bisherigen Features bietet die Version 1.10 folgende Neuheiten:

- Seperate Compilation, kleine Applikationen durch einen Linker; 68030/68882-Assembler/Disassembler; History-Buffer für die Kommandozeile; eingebundener Streamfile-Editor, volle Unterstützung des TTs...

Natürlich bietet bigFORTH auch sonst alles, was ein gutes FORTH so braucht:

- **Mächtiger Compiler:**
  32-Bit-System, Native-Code, voll relokatibel, weitgehend FORTH-83-kompatibel, nutzt Screen- und Streamfiles; relokatible Turnkey-Applikationen, denen man FORTH nicht mehr ansieht, sind problemlos machbar...
- **Vielseitige Tools:**
  68030-Assembler/Disassembler, Multitasker, sourcefähiger Decompiler, Source-Level-Debugger (Single-Step und Trace, beliebig viele Breakpoints), Post-Mortem Dump und Returnstack-Trace bei allen Fehlern, resetfest (für den Ausstieg aus der Endlosschleife), Multiwindow-Editor für Screen- und Streamfiles (natürlich unter GEM), beliebiger anderer Editor nutzbar...
- **Umfangreiche Libraries:**
  Komfortables Fileinterface, Druckertreiber, vollständige Betriebssystemanbindung (GEMDOS, BIOS, XBIOS, GEMAES, GEM-VDI und Line-A-Grafik), Floating-Point-Arithmetik, leistungsstarkes Memory Management, High-Level-GEM-Library, Turtle Graphic und mehr...
- **Transparenz:**
  Sämtliche Sourcen (einschließlich Kernal) auf Diskette, das modulare System ersetzt den Target-Compiler...

Die ausführliche deutsche Dokumentation (über 250 Seiten Handbuch) wird Anfängern ihre ersten Schluckbeschwerden bei dem großen Bissen beseitigen, aber auch für alte Hasen von Vorteil sein...

Und das alles nur für[1] DM 200.-

Update von Version 1.00: DM 20.-

Für Vorsichtige gibt es eine frei kopierbare Demoversion (ohne Sourcen&Handbuch) für eine Schutzgebühr von DM 10.-

Bestellungen an:

Bernd Paysan,
Stockmannstr. 14
D-8000 München 71
Tel. 089/798557

[1](alle Preise incl. Mehrwertsteuer und Versand)

Wunschtraum eines Editors

Mein Wunschtraum an die Autoren der "Vierten Dimension" ist folgendes:

Die Arbeiten sollenn enthalten:

Titelseite, Zusammenfassung, Schlüsselworte, Text, Quellenangaben, Illustrationen, Tabellen und Qellcode. Alles zusammen auf Datenträger in einzelnen Files mit Inhaltsangabe wo und was zu finden ist.

TITELSEITE:
Eine eigene (erste) Seite, Titel, Autor und Institut oder Betrieb, in dem oder für den die Arbeit angefertigt wurde.

ZUSAMMENFASSUNG:
Die Zusammenfassung von 50 bis 100 Worten sollte Absicht, Methoden, Ergebnisse und Schlußfolgerungen der Arbeit enthalten und für jedermann verständlich sein.

SCHLÜSSELWORTE:
Etwa fünf Schlüsselworte sollten ausgesucht werden, für die die Arbeit relevant ist.

TEXT:
Wenn möglich sollte der Text in Zeitungsform aufgebaut werden. Das wichtigste zuerst. D.h. in einer kurzen Einleitung über das Ziel der Arbeit informieren. Als Hilfe gibt es im Zeitungswesen die fünf verschieden "W's", WER, WO, WIE, WAS und WARUM. Desweiteren Materialien, und Methoden hinreichend genau wiedergeben, über die Entwicklung der Ergebnisse oder Systeme berichten, diese diskutieren und Schlüsse ziehen.

DANK:
Für Hilfen oder Rat, technische Mitarbeit, Materialen usw. sollte Dank in einem eigenem Abschnitt der Arbeit angesprochen werden.

QUELLENANGABEN:
Die Quellenangaben sollen am Ende des Textes stehen, alphabetisch nach Autoren geordnet und durchnummeriert [in eckigen Klammern] sein. Im Text beziehen sich die [Nummern] auf diese Liste. Zeitungsartikel sollen mit Namen und Initialen von allen Autoren, dem vollen Titel des Artikels, des Namen der Publikation, der Rubrik, dem Erscheinungsjahr und der Nummer der ersten und letzten Seite des Artikels genannt werden.

ILLUSTRATIONEN:
Sollten unbedingt auf das Notwendigste beschränkt werden und keine Daten enthalten. Diagramme und Kurvenverläufe bitte alle im Zeitalter der Elektronischen Kommunikation auf Datenträger schicken. Anleitungen derselben sollen auf einem eigenen File gespeichert und entsprechend gekennzeichnet werden, damit der Editor nicht erst lange suchen muß.

TABELLEN:
Für Tabellen gilt dies ebenso.

QUELLCODE:
Bitte auch in einem eigen File, im ASCII-Format, 64 Zeichen breit. Andere Formate ergeben Probleme und können nicht eingespiegelt werden. Diese werden an den Absender zurückgeschickt.
Der Code soll eine Seiten- und Screen-Nummer enthalten, eine Überschrift tragen und im Text angesprochen werden. Die Kommentare zum Code sollen zeigen, WAS der kommentierte Abschnitt ausführt, also den Sinn wiedergeben.
Desweiteren sollte der Quellcode nicht mehr als "neun" Screens oder Blocks enthalten, und wenn doch, nur in dreier Schritten, damit das Layout auch weiterhin möglichst konstant aussieht.

ANMERKUNG:
Wer seine Beiträge überarbeiten lassen will, sollte dies kenntlich machen. Der redaktionelle Beirat steht immer zu Diensten. Bei zu langen Texten besteht die Gefahr, daß diese gekürzt werden. Dies fällt nicht, die Erfahrung hat es gelehrt, immer zur Zufriedenheit der Autoren aus. Auf jeden Fall werden die korrigierten Beiträge mit Ausnahme der Codes zur Kontrolle zurückgeschickt. Und nur mit gegenseitigen Einvernehmen veröffentlicht. Also nicht gleich entmutigen zu lassen. Denn "immer nach der Devise, "Wer hier schreibt, blamiert sich nicht".

Zu erreichen ist die REDAKTION und das FORTH-Büro am sichersten auf postalischem Wege:

REDAKTION
Peter Dinies
Metzstraße 38
W-2300 Kiel 1
Tel. 04 31-1 32 39

oder

FORTH-Büro:
FORTH-Gesellschaft e.V.
Postfach 1110
W-8044 Unterschleißheim
Tel. 089-317 37 84
DFÜ 08841-5880

Ich hoffe, daß sich dies verwirklichen läßt und mir damit einige Arbeit erspart wird.

Euer Editor Peter Dinies

Echtzeit '91

Kongreß und Ausstellung

Am 11. bis 13. Juni findet die Messe "Echtzeit '91" in Sindelfingen bei Stuttgart statt. Der Kongreß und die Aussstellung für angewandte Echtzeit-Datenverarbeitung in Automation, Meßtechnik, Simulation und anderen technisch/wissenschaftlichen Bereichen beeinhaltet Anwendungs-orientierte Themen und geplante Vortragsreihen.

Prozeßdaten-Erfassung und Verarbeitung

- Prozeß-Steuerung
- Leit-Technik
- Anlagen-Steuerung
- Meßdaten-Erfassung und -Verarbeitung
- Echtzeit-Bildverarbeitung
- Simulation und weitere typische Echtzeit-Anwendungen aus Technik und Wissenschaft

Geplante Vortragsreihen:

- Architektur von Mehrprozessorsystemen
- Kommunkikationssysteme für die Fertigung
- Anwendung in der Prozeß technik
- Echtzeit-Betriebssysteme
- Neuartige Architekturen für Echtzeit-Anwendungen
- Entwicklungswerkzeuge (Tools) Programmiersprachen
- Das FORTH-Konzept
- Anwendungen in Meß- und Steuerungstechnik
- Neuartige Echtzeit-Systeme in der Praxis
- Hardwarekomponenten für Echtzeit-Systeme
- Echtzeit-Betriebssystemkerne im Einsatz
- Simulation
- Systemkomponenten
- Fallstudien
- Systemtechnische Lösungen
- UNIX für Echtzeit-Anwendungen
- PC und Echtzeit
- Produkte für die Software-Entwicklung

Veranstalter des Kongresses sind die FORTH-Gesellschaft e.V. unter Mitwirkung des PEARL e.V. in Kooperation mit

- dem Arbeitskreis Echtzeit
- Mitglied des ADA Deutschland
- ACM (Association for Computing Machinery, New York)
- VITA (VMEbus Internatioal Trade Association)

Weitere Informationen sind vom Organisator von Kongreß und Ausstellung

Dipl. Ing. Ludwig Drebinger
Agentur für technische Fachkongresse

Destouchestraße 16
W-8000 München 40

Tel. 089-333 0 333
FAX 089-33 27 61

zu erhalten.

Computer-Club

an der RWTH-Aachen e.V.

Arbeitsgruppe

An alle,
die Forth kennen oder kennenlernen wollen,
die Forth-Probleme mit anderen besprechen
und gemeinsam lösen wollen, die am Aufbau eines
Forth-Applikations-Pools mitarbeiten wollen,
die Forth bei Institutsprojekten verwenden!
Für all die gibt es jetzt eine
Anlaufadresse:

Seminargebäude der RWTH-Aachen, Raum 214

Wir treffen uns
montags zwischen 20 und 22 Uhr.

Haupttermin:
jeder erste Montag des Monats

Mitgliederlisten - wozu und wie?

Etwas in Vergessenheit geraten sind vielleicht die Mitgliederlisten der Forth Gesellschaft. Sie erscheinen als Beilage im letzten Heft eines Jahrgangs. Damit können Verbindungen aufgenommen werden. Wir sortieren deshalb regional nach Postleitzahlen. Um dem Adressenhandel vorzubeugen, wurden bisher keine Anschriften veröffentlicht sondern nur die Telefonnummern. Ein Blick in die letzte Liste zeigt aber hinter erstaunlich vielen Namen gar keine Telefonangabe. Da wollten oder konnten einige keine Angaben machen.

Macht nichts. Über das FORTH-büro kann man trotzdem Kontakt aufnehmen. Die übrigen Angaben zum benutzten Rechnertyp, Forthsystem und Interessengebiet waren unter Experten schon immer umstritten - sie sollen die Wahl erleichtern. Obwohl das recht nebensächlich ist, wie ich finde - zumal da wohl fast nichts stimmt. Oder haben Sie ihre Daten dort immer auf den neuesten Stand gebracht? Ich habe dem Forthbüro noch nicht verraten, daß der alte Apple-II jetzt in der Museumskiste verpackt im Keller liegt, mittlerweile ein Toshiba Laptop meine Schreibmaschine ersetzt hat und ein AT-Tower mein Büro ziert und FPC Einzug gehalten hat. Ja, ja - drei Jahre liegen erst dazwischen. Über diese Einträge sollten wir in München mal gemeinsam nachdenken.

Michael Kalus

Administration

Bitte benachrichtigen Sie das FORTH-Büro rechtzeitig bei Änderungen Ihrer Anschrift oder Bankverbindung, da fehlgeleitete Postsendungen und Überweisungen unnötige Kosten verursachen.

Der VIERTEN DIMENSION Volume VI, No. 4, lag ein Mitgliedsantrag bei. Teilen Sie uns darauf Änderungen für Ihren Eintrag in der Mitgliederliste mit oder benützen Sie diesen Antrag zum Werben neuer Mitglieder.

Beachten Sie bitte, daß die ermäßigten Beiträge nur gegen Nachweis gewährt werden, also für 1991 die Studienbescheinigungen etc. nicht vergessen.

Für weitere Fragen stehen wir Ihnen gerne zur Verfügung.

Telefonisch erreichen Sie uns unter 089-317 37 84.

In eigener Sache

Da war es wieder, das alte Problem mit der letzten Ausgabe eines Jahrgangs. Vorgesehen für Dezember, erschien Heft 4/90 nun im Januar'91 mit über vier Wochen Verzögerung.

Eigentlich war alles wie geplant fertig. Dann brauchte die Post innerhalb von München für das Paket mit dem Layout bis zum Drucker sechs Tage. Dessen Weihnachtsurlaub war darüber inzwischen angebrochen. Doch im neuen Jahr dann war es soweit: Montag früh am 14. Januar wurden alle Hefte abgeschickt. Meine Ausgabe hatte ich immerhin schon am Samstag den 20. Januar in meinem Briefkasten- fünf Tage darauf.

Die "Schneckenpost" der Deutschen ist also ein Problem, ein anderes ist der Termin nahe an Weihnachten. Da läuft an vielen Stellen offenbar nichts mehr. Immer hapert es mit dem letzten Heft. Machen wir also aus der Not eine Tugend und lassen die Hefte zu Beginn des Quartals erscheinen: Januar, April, Juli, Oktober. Mal sehen, ob wir die Zeit bis Oktober einholen können, meint

Michael Kalus

...In letzter Minute....In letzter Minute....

Subject:
Division... Is both types of division in BASIS arithmatically wrong???
From comp.lang.forth Tue Mar 19 11:21:49 1991
From: S47852EF@ETSUACAD.BITNET ("Frank C. Earl")
Newsgroups: comp.lang.forth

To the net:

This is an emprical proof of the fact that the forms of division used in most of the current computer languages (Including the two used in the BASIS...) are incorrect in the results given for the remainder for operations in the third quadrant (both divisor and dividend are negative). This is in response to a comment made to me by Mitch Bradley over a statement that I made about this being a problem in both
types of division.

Passed to /MOD is the values minus four and minus three-

Results:

```
F-PC          1 -1
MVP           1 -1
TaskForth     1 -1
```

So, you might ask, what is wrong with this?

The remainder is NEGATIVE. This is incorrect, as you will soon see...

Consider,

$$-4\ -3\ /MOD \text{ is equivalent to } \frac{-4}{-3}$$

in fractional expression terms.

$$\text{Given this, } \frac{-4}{-3} = \frac{-1}{-1} * \frac{4}{3}. \quad \text{The } \frac{-1}{-1} = 1$$

because the signs cancel.

Therefore, our calculation is really with

$$\frac{4}{3}, \text{ which is } 1 + \frac{1}{3}.$$

This is what /MOD passes back as a result - the whole number resulting from the division and the dividend of the fraction that's left over.

But, every Forth dialect that I have asked, tells me that it's

$$1 + \frac{1}{3},$$

which, from what I just said is *quite* wrong... (Not only is Forth off here, but every language dialect I've tried does the same thing (ADA's the only one I know of that does the right calculation for the remainder portion)). If you think that this doesn't matter- think again...

$$1 + \frac{1}{3} = \frac{4}{3} \text{ and, } 1 + \frac{-1}{3} = \frac{1}{-3} = \frac{2}{-3} \text{ and,}$$

$$\frac{4}{3} \text{ does not equal } \frac{2}{3}.$$

That's sufficient enough an error to justify re-working it (What if you don't check to see that this throws off your calculations- it could prove catastrophic- might cause a horrible accident of some sort if it was used in a critical application that only very rarely drifted off into the third quadrant??) and looking at ways of doing it better...

The next posting from me (time permitting) will be a description of the algorithm I propose should be considered for standardization (I know it's too late- but we could start considering it for the next standard...) and some sample Forth-83 (F-PC) code that does the desired operation...

Frank

Subject: Modulus
From comp.lang.forth Tue Mar 19 11:21:58 1991
From: wmb@MITCH.ENG.SUN.COM (Mitch Bradley)
Newsgroups: comp.lang.forth
Organization: The Internet

This is an emprical proof of the fact that the forms of division used in most of the current computer languages (Including the two used in the BASIS...) are incorrect in the results given for the remainder for operations in the third quadrant (both divisor and dividend are negative). ...

Consider,

$$-4\ -3\ /MOD \text{ is equivalent to}$$

$$\frac{-4}{-3} \text{ in fractional expression terms.}$$

Sorry, this isn't correct. The mathematical definition of signed integer division with remainder is the following equation holds:

```
quotient*divisor + remainder = dividend
```

There are several ways of arranging the signs to make this work out correctly; the mathematical "mod"function arranges the signs so that that, if the remainder is nonzero, its sign is the same as the divisor's.

Reference: Knuth "The Art of Computer Programming", Vol. 1.

There are other choices, but the Forth-83 choice is definitely mathematically correct, and there is considerable evidence to suggest that it is the best choice for many if not most numerical applications. Robert Berkey has written extensively on this topic, and rather than recap his arguments, I refer interested readers to the literature.

Following up on the example presented, let's recast it in fractional form:

```
                  remainder   dividend
quotient    +     --------- = --------
                   divisor     divisor

                    -1          -4
   1        +     --------- = --------
                    -3          -3
```

Thus we see that the error in the original argument stems from the fact that, which the signs of the dividend and divisor cancel out, the fractional representation implies that the remainder is also divided by the divisor. In order for the divisor's sign to be canceled in the remainder term, the remainder must be negative.

Mitch

Subject: Re: Modulus
From comp.lang.forth Tue Mar 19 11:22:05 1991
From: S47852EF@ETSUACAD.BITNET ("Frank C. Earl")
Newsgroups: comp.lang.forth
PST Mitch Bradley said:

Sorry, this isn't correct. The mathematical definition of signed integer division with remainder is the following equation holds:

```
quotient*divisor + remainder = dividend
```

But it *is* correct- a fractional representation is simply what you just said

From what I was told from 1st and second grade on, a fraction is the same thing a division operation and that integer division returns the whole number result and the divisor of the left over fraction, or in other words-

```
   4
   -  = 4 / 3 = 1.33333
   3

which is approximately equal to

         1
   1 + - .
         3
```

I get the SAME results from a calculator- ANY calculator.

Also, all the calculators in my house give me the *SAME* result for -4 /-3. If, I'm wrong, then ALL the calculators we use are also wrong... (I don't think anybody wants to touch that w/ a 10-ft pole... :)

Following up on the example presented, let's recast it in fractional form:

```
                  remainder   dividend
quotient    +     --------- = --------
                   divisor     divisor

                    -1          -4
   1        +     --------- = --------
                    -3          -3
```

Your example didn't come ove the net well at all... Please re-post it if you'd like. It really doesn't show any errors in my reasoning at all in the form that I got...

BTW- One of my grad instructors has checked my reasoning and he agrees; this coming from a math instructor, has me thinking that something's awry...

Thanks, Frank

Subject: Conditional compilation
From: wmb@MITCH.ENG.SUN.COM (Mitch Bradley)
Newsgroups: comp.lang.forth
I'm curious to know if "common use" is isomorphic with "LMI does it" in this case or if there really are other users.
(As I pointed out, Upper Deck Systems does it differently).

I don't know of any systems other than LMI using the names ".IF .ELSE .THEN".
In fact, I don't know of any two systems that agree on ANY set of names.
The only set of names that has much chance of multi-system use is IFTRUE .. OTHERWISE .. IFEND , and that set has the unfortunate property that OTHERWISE is a well-formed English word, and thus tends to appear in text comments, confusing the IFTRUE parser.
Everything else being equal, I would go with LMI over Upper Deck Systems, since I suspect that LMI has at least 10 times as many users as Upper Deck Systems (this is not intended as a criticism of UDS).
I think that naming consistency is important, but I don't think it is as important as having the functions you need. The naming thing is an annoyance, but lack of functionality can keep you from doing something you need to do.
If a convenient opportunity arises, I will propose a name change, but I am worried that raising the issue will open the door for the minimalists to propose deleting the functions entirely. I don't want that to happen.
Mitch

Subject: Conditionals !
From: ForthNet@willett.pgh.pa.us
(ForthNet articles from GEnie)
Newsgroups: comp.lang.forth
Category 10, Topic 4 Message 3
B.RODRIGUEZ2 [Brad] at 10:22 EST
I don't know of any systems other than LMI using the names
".IF .ELSE .THEN".
Klaus Schleisiek-Kern of DELTA t uses these names.
I am worried that raising the issue will open the door for the minimalists to propose deleting the functions entirely. And I am worried about functions being railroaded into the BASIS. Is it the TC's policy to put things in at the last minute so as to avoid discussion of their removal?

- Brad

--FORTH-Gruppendynamik--

FORTH-Gruppen:

W-1000 Berlin
Claus Vogt
Tel. 030/2 16 89 38
Treffen nach Absprache

W-4130 Moers 1
Friederich Prinz
Tel. 02841/ 5 83 98
Treffen nach Absprache

Gruppe Rhein-Ruhr:
Jörg Plewe
Tel. 0208/42 35 14
Treffen jeden ersten Samstag im Monat, Münsterstraße 199, W-4000 Düsseldorf, Gebäude des S-Bahnhof Derendorf

W-6104 Seeheim-Jugenheim
Andreas Soeder
Tel. 06257/27 44
Treffen an der VHS, an einem Mittwoch in der Mitte des Monats

W-6800 Mannheim
Thomas Prinz
Tel. 06271/28 30
Ewald Rieger
Tel. 06239/86 32
Treffen jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz, Mannheim-Neuostheim

W-7000 Stuttgart
Wolf-Helge Neumann
Tel. 0711/88 26 38
Treffen nach Absprache

O-Leipzig
FORTH-Grupppe Leipzig:
Michael Balig, Lützner Plan 17, O-7033 Leipzig
Dr. Jürgen Hesse
Tel. 041//69 56 02
Liselotte-Herrmann-Str. 40, O-7050 Leipzig

FORTH für Ratsuchende:

Jörg Staben
Tel. 02103/5 56 09
dienstags und freitags von 20.00-22.00 Uhr

Frank Stüss
Tel. 06187/9 15 03

Karl Schroer
Tel. 02845/2 89 51

Andreas Findewirth
Tel. 05221/2 35 04

Andreas Jennen
W-1000 Berlin, UUCP

Jörg Plewe
Tel. 0625/42 35 14

FORTH Fachgruppen:

W-6800 Mannheim
FIS (FORTH Integriertes System) - Datenbank, Textverarbeitung, Kalkulation
Postadresse:
Dr. med. Elemer Teshmar, Danziger Baumgang 97, W-6800 Mannheim 31

FORTH Interessengebiete:

volksFORTH/ultraFORTH
Klaus Kohl
Tel. 08233/3 05 24
Klaus Schleisiek-Kern
Tel. 040/2 20 25 39

32-Bit Systeme
Robert Jones
Tel. 02434/45 79

Künstliche Intelligenz
Ulrich Hoffmann
Tel 0431/67 88 50

NC4000 Novix Chip
Klaus Schleisiek-Kern
Tel. 040/2 20 25 39

Relationale Netze, Künstliche Intelligenz Realtime
Wigand Gawenda
Tel. 040/44 69 41

Gleitkomma-Artihmetik
Andreas Döring
Tel. 07 21/5939 35

32FORTH
Rainer Aumiller
Tel. 089/6 70 83 55

PostScript/FORTHscript
Christoph Krinninger
Tel. 089/ 7 25 93 82

FORTH im Unterricht
Rolf Kretzschmar
Tel. 02401/43 90

Objekt-orientiertes FORTH
Christoph Krinninger
Tel. 089/7 25 93 82
Ulrich Hoffmann
Tel. 0431/67 88 50

F-PC Zimmer FORTH
ASYST, Meßtechnik
Arndt Klingelnberg
Tel. 02404/6 16 48
box:geo1:klingelnberg

FORTH Gruppengründung:

W-3300 Braunschweig
Martin Holzapfel
Tel. 05 31/35 12 62

FORTH Fachgruppengründung:

W-7000 Stuttgart 80
Jörg Tomes
Tel. 07 11/7 80 22 93
nur am Wochenende

FORTH-SYSTEME

ANGELIKA FLESCH

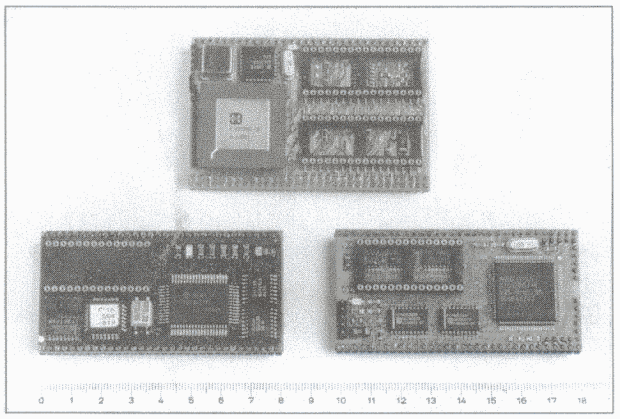
Postfach 1103
D-7814 Breisach

Telefon (0 76 67) 5 51
Telefax (0 76 67) 5 55

UR/FORTH

- Forth-83 Standard
- Für MS-DOS, OS/2, 80386
- Direkt gefädelte Code Implementationen mit dem obersten Stackwert im Register um größtmögliche Ausführungsgeschwindigkeit zu erreichen
- Segmentiertes Speichermodell mit Programm, Daten, Headers und Dictionary Hash Table jeweils in einem getrennten Segment
- Komplett gehashtes Dictionary führt zu extrem schneller Übersetzung
- Mächtige neue String Operatoren (Suche, Extraktion, Vergleich und Addition) sowie einen dynamischen String-, Speichermanager
- Kann mit Objektmodulen, die in Assembler oder anderen Hochsprachen erzeugt wurden, gelinkt werden
- Native Code Optimizer zur direkten Umsetzung in 80 x 86 Code im Lieferumfang

ModuNORM



CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z. B. 6303
- 16 Bit z. B. V25
- Highspeed RTX-2000/1
- Softwareunterstützung durch SwissFORTH™
- Thermodrucker und Kontroller

Bitte fordern Sie unseren Produktkatalog und Preisliste an. FORTH-Gesellschaftsmitglieder erhalten bis zu 10 % Rabatt (artikelabhängig).

LMI FORTH-83 Metacompiler

Der LMI Forth Metacompiler wird mit komplettem Quellcode für ein ausführlich ausgetestetes, Hochgeschwindigkeits Forth 83 Kern ausgeliefert, wobei Sie die Auswahl aus folgenden Zielprozessoren haben:

8086/8088	8096/97
Z80	HD64180
8080/8085	8031/32/535
68000	6303
Z8	6502
1802	V25
6809	68HC11
65816/65802	RTX 2000

Sie erzeugen schnelle und kompakte Anwendungen, indem Sie Ihre Quellprogramme mit unserem Forth Nucleus zusammenstellen und ihn mit dem LMI Forth Metacompiler übersetzen.

Forth Programme, die mit einem LMI interaktiven Forth System z. B. PC/FORTH oder Z80 Forth geschrieben und getestet wurden, werden im Normalfall mit nur geringen Änderungen übersetzt.

Serieller ROM/RAM Simulator

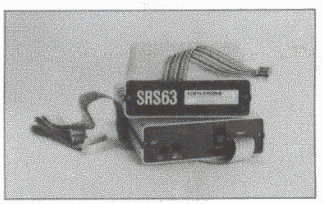
Entwickeln Sie romfähige Programme ?

Müssen Sie neu entwickelte Einplatinencomputer testen ?

Setzen Sie 2764, 27128, 27256, 27512 oder 4364, 43256 oder kompatible ROM/RAM-Bausteine ein ?

Wollen Sie diese Bausteine mit bis zu 38 400 Baud über die serielle Schnittstelle laden ?

Können Sie eine zusätzliche serielle Schnittstelle über den Speichersockel zum interaktiven Programmieren gebrauchen ?



Dann ist unser SRS63 die optimale Ergänzung Ihres Arbeitsplatzes.

Sie werden vom Preis-Leistungsverhältnis überrascht sein.

Unsere ROM-Compiler liefern direkt verwendbare Dateien, wir akzeptieren auch Intel-Hex oder Motorola-S-Formate.

mikrap

Mikrap
AG für Mikroelektronik-Applikation

CH-8840 Einsiedeln
Langrütistrasse 44
Postfach

Telefon 055 53 65 55
Telefax 055 53 61 23