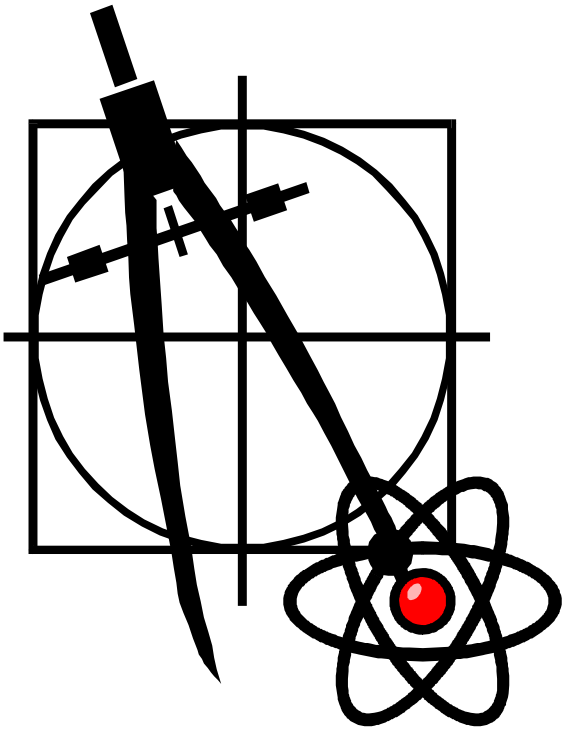
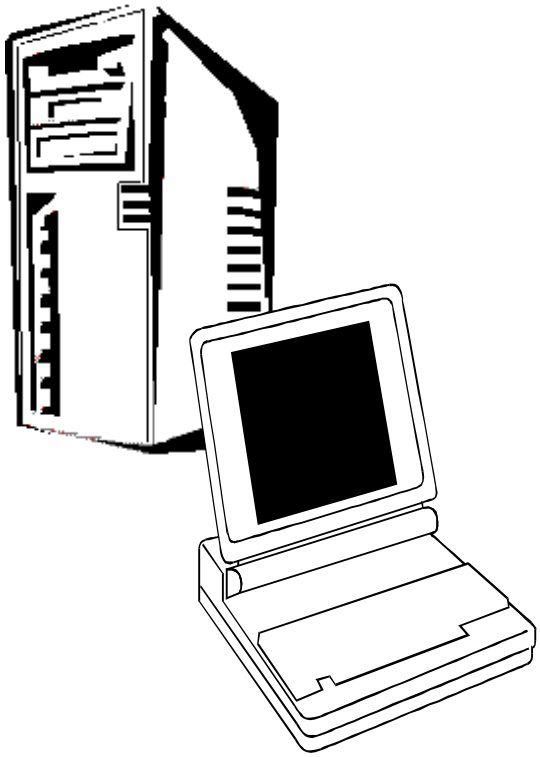
VIERTE DIMENSION

2 / 2001

Das FORTH-Magazin

**für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten.**





In dieser Ausgabe

17. Jahrgang - **Ausg. 2/2001** – 10,-DM

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

Inhaltsverzeichnis

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 16 12 04
D-18025 Rostock
Tel.(Anrufbeantw.): 0381-400 78 28
Fax: 0381-400 78 28
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto. 563 211 208

Redaktion & Layout (vorübergehend)

Martin Bitter
Möllenkampweg 1a
46499 Hamminkeln
Tel.: 02857-1419
E-Mail: VD@FORTH-EV.DE
mbitter@bigfoot.de

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß 2001

März, Juni, September, Dezember
jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

DM 10,- zzgl. Porto u. Verp.

Meeresrauschen und schalltote Räume

So! Geschafft!
Manchmal treten Aliens aus dem Weltall mit mir in Verbindung. Sie bedienen sich dazu eines geliehenen Laserdruckers, der eigentlich die Vierte Dimension ausspucken soll. Wahrscheinlich liegt es an jener seltsamen 'Vierten Dimension' – jedenfalls sorgen diese Aliens dafür, dass nicht das Erwartete auf dem Papier erscheint, sondern *Geheimzeichen,* die ich nicht entziffern kann. Selbst Bernd Beusters Kenntnisse (vgl. S.8) helfen da nicht weiter. Oder wirft mich die 'Vierte Dimension' zurück an den Anfang aller Zeit, als es noch keine ordentlichen Druckertreiber gab?
Aber immerhin: Sie (die Aliens) wollen zu mir Kontakt aufnehmen!

Akustiker, einige Wahrnehmungspsychologen und andere glückliche Menschen kennen dieses Erlebnis: In einem schalltoten Raum befällt den 'normalhörigen' unwillkürlich ein Gefühl der Beklemmung. Besonders wenn man versucht sich zu unterhalten oder einfach nur mal eben so laut zu schreien. Die vielfältigen stets vorhandenen kleinen Umweltgeräusche, der leise Hall der verschiedenen Stimmen und Klänge an akustisch reflektierenden Flächen, all das nehmen wir unbewusst war und benutzen es zur Orientierung im Hier und und Jetzt.
Gerade das Fehlen dieser 'unterschwelligen' Klänge erzeugt das Gefühl des Unwohlseins.
Hugo Kückelhaus hat in seinem 'Erfahrungsfeld zur Entfaltung der Sinne' einen 'Brüllstein' geschaffen. Das ist ein Felsblock mit einer Öffnung, die sehr effektiv den Schall der hineingebrüllten eigenen Stimme verschluckt – und da ist es wieder dieses Gefühl der Irritation., diesmal künstlerisch-philosophisch überhöht.
Ähnlich irritiert fühlt sich ein Editor, wenn 'seine' Zeitschrift, die im Kern das Kind eines ganzen Vereines ist, sehr wenig Reaktionen hervorruft (vgl. Leserbriefe).

Hält irgendwer, das kann auch eine Maschine sein, ein kleines Gefäß, vielleicht eine Muschel ans Ohr oder eben ans Mikrophon, so hört man es rauschen. „Das Meer rauscht!“ erzählte man uns, als wir noch Kinder waren, später hieß es dann aufklärerisch „Das ist dein eigenes Blut.“ noch später ganz wissenschaftlich „Es sind die 'unhörbaren' Umweltgeräusche, die durch die Resonanz des Gefäßes über die Hörschwelle gehoben werden, die man da hört.“
Das läßt sich leicht überprüfen: In einem schalltoten Raum, rauscht eine Muschel nicht!

Meinen Kampf gegen die Aliens habe ich gewonnen, sonst könnten Sie nicht darüber lesen – ob es in der Muschel (wieder kräftig) rauscht, das liegt an Ihnen, an den Mitgliedern und sonstigen Lesern der Vierten Dimension, an Ihrem Widerhall!

So genug des 'pädagogischen Zeigefingers' (ich kann es eben nicht lassen!).
Ich wünsche Ihnen allen viel Spaß beim Lesen dieser Vierten Dimension und der nächsten und der nächsten und der ...

may the forth be ... M. Bitter

(Interimeditor)

PS. Leider war mein privater Sternenkrieg recht langwierig, so dass nicht mehr garantiert ist, dass diese Vierte Dimension den Weg zu ihnen über Mehrhoog, Mittweida und Rostok noch rechtzeitig vor Beginn der Jahrestagung schafft.
Aus diesem Grund wird das notwendige Informationsmaterial den Tagungsteilnehmern und Teilnehmerinnen mit separater Post zu gesandt.

Quelltext Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse.

MB

.Auf die letzte Ausgabe der Vierten Dimension hin gab es einen (sic!) Leserbrief, aus dem sich ein zwar kurzer aber intensiver Gedankenaustausch zwischen Ulrich Paul und mir (M.B.) ergab. Alles war (ist) hochinteressant -aber aus Platzgründen können hier nur Auszüge wiedergegeben werden. Ich möchte auf diesem Weg dazu ermutigen, dem Beispiel von Herrn Paul zu folgen.

Leserbrief:

Etwas ganz anderes: Ich bin ja nur noch ein passives Mitglied der FG, da ich – ehrlich gesagt – das ganze Geplappere, daß Forth ALLES kann was die anderen auch können und das auch noch besser oder schon länger, einfach idiotisch finde. Ich halte jeden für einen Schwachkopf, der behauptet, daß man StarOffice einfacher in Forth als in C schreiben hätte können. Unsinn! Dafür war Forth nie gedacht und es ist bis heute nicht in der Lage, auch bloß annähernd in diesen Bereich vorzustoßen. Seine Stärken sind hier seine Schwächen! Forth ist prädestiniert für Rapid-Prototyping, Proof-of-Concept und embedded Solutions. Hier kann es seine Vorzüge ausspielen. Aber wenn ich mit Target-Compiling anfange, dann kann ich gleich einen optimierenden Compiler für das Zielsystem einsetzen und in C (oder was auch sonst immer) verwenden. Die Libraries dieser Compiler sind gut, getestet und standardisiert.
Ich war echt erfreut in der VD kritische Artikel zu diesem Thema zu finden. Vor allem, daß sie von so „alten Hasen“ stammen. Ja, die Forth-Gemeinde muß umdenken und sich neu positionieren. Was bestimmte Bereiche betrifft ist der Zug für Forth abgefahren und den holen wir nie wieder ein.
Konzentrieren wir uns doch auf das, was Forth zu Forth macht:
1.) Die kompakte Entwicklungsumgebung mit Interpreter und Compiler

2.) die leichte Portierbarkeit

3.) die CREATE – DOES Konstruktion

4.) die einfache Behandlung von Hardware

5.) der moderate Speicherbedarf.

Viele Sprachen bieten eine oder mehrere der o.g. Eigenschaften, aber alle zusammen eben nur Forth. Was Forth nicht bietet:
1.) standardisierte Bibliotheken

2.) einen Linker zu Routinen in anderen Sprachen oder um Forth-Routinen in anderen Sprachen zu verwenden

3.) eine ausgefeilte Oberfläche

4.) einen optimierenden Compiler (mit allen heute üblichen Features!) Das bißchen Optimierung, das Forth-Compiler machen, ist einfach Pippifax im Vergleich zu dem heutigen Standard.

5.) Checker auf semantische Fehler (falscher Typ und so) Was heißt das?

Das heißt: Wir müssen 2 Wege gehen. Der eine führt uns dahin, daß man so einfach Forth-Routinen mit Programmen in anderen Programmiersprachen zu einer Anwendung linken kann wie es bei den anderen untereinander schon lange üblich ist. Auch anders herum muß es gehen: Daß man in seinem Forth-Programm auf Routinen zugreifen kann, die in anderen Sprachen geschrieben wurden. Auf Deutsch: Ein Object-Code-Standard muß her und der muß eben der Standard sein, an den die anderen sich schon lange halten. Kein Extra-Süppchen wie üblich!
Der andere Weg führt uns dahin, daß Forth-Programmierer so einfach Code von anderen Forth-Programmierern übernehmen können wie bei den anderen üblich. Das ist nahe verwandt mit dem oben gesagten, muß aber nicht notwendigerweise so implementiert werden. Hier könnte sich durchaus ein eigener Standard bewähren. Zwei Punkte sind hier besonders wichtig: Erstens muß der Standard es sicherstellen, daß der Fremd-Code auf jedem System gleich läuft und zweitens muß es möglich sein den Fremd-Code in einer Form einzubinden, in der die Urheberrechte des Autors gewahrt werden – also nicht im Sourcecode!
Glaubst Du, daß das jemals kommt? Ich bin mir sicher, daß es NICHT kommt. Was glaubst Du, was passiert, wenn jemand auf die Idee kommt, etwas gleichartiges wie CREATE – DOES in C oder einer anderen gleichwertigen Sprache einzubauen? Dann ist Forth tot, mausetot! Denken wir daran: die umgekehrte polnische Notation ist KEIN Alleinstellungsmerkmal von Forth! Es macht den Interpreter und den Compiler einfacher, aber das ist auch schon alles.
Wenn ich eine Maschinensteuerung implementieren muß, die eine Kommandozeilen-Schnittstelle haben muß, dann werde ich der heutzutage vielleicht noch das Feeling von Forth geben, aber auch nur dann, wenn der Speicher wirklich knapp ist. Stehen mir ein paar Kilobytes mehr zur Verfügung, dann ist die RPN weg! Dann schaut das eher wie Basic aus. Damit kommen die Leute einfacher klar und ich kann semantische Checks einbauen, da sie durch explizite Klammerung und ähnliche „Restriktionen“ gezwungen sind ihre Absicht klarer auszudrücken. Der Stack wäre komplett unsichtbar und unzugreifbar! Nix geht dann mehr, was die Kiste so einfach in‘s Nirwana schickt wie „0 EXECUTE“.
Würde ich der Maschinensteuerung einen Compiler spendieren? Nein! Der würde auf dem Laptop laufen mit dem ich sowieso auf die Steuerung zugreife. Das fertige Kompilat geht runter zur Steuerung. Alle Checks und Arbeit mache ich dort, wo mir massig Speicher und Rechenleistung zur Verfügung steht. Und wo ist jetzt noch Forth? Futsch! Der Anwender weiß nicht in welcher Sprache ich die Steuerung programmiert habe. Er sieht nur das, was ich ihm als Oberfläche anbiete und das muß halt so ausschauen, daß der Anwender damit klar kommt. Ob der freiwillig sich für Forth entscheidet? Nö! Täte ich ja auch nicht unbedingt.
Ist Forth für mich tot? Lasse ich besser die Finger davon? Zweimal nein! Ich schreibe öfter Treiber, früher unter DOS, jetzt unter Linux. Wenn ich die neue Hardware nicht flott in den Griff kriege, dann werfe ich Forth an und beäuge sie näher um herauszufinden, was ich nicht richtig verstanden habe. Aber der Treiber wird in C geschrieben – komplett. Und wenn ich einmal wieder auf die Schnelle einen wirren Konverter von einem Datenformat auf ein anderes brauche, dann werde ich wahrscheinlich das auch mit Forth erledigen. Oder wenn ich

ein Protokol, das ich mir ausgedacht habe, erst einmal simulieren will, dann ist Forth dran. Aber viel weiter geht meine Motivation nicht mehr. Dann wird Forth zu unhandlich. Der Gewinn durch die direkte Art von Forth wird dann überschattet von der mickrigen semantischen Unterstützung des Compilers. Aufwendigeres schreibt man besser in C oder Java oder so.
Warum stellt sich eigentlich die Forth-Gemeinde so stur? Klar, die Forth-Gesellschaft hat als erklärtes Ziel die Verbreitung von Forth zu fördern. Aber kann man das nicht auf „die Verbreitung von Programmierkenntnissen allgemein und unter besonderer Berücksichtigung von Forth im speziellen“ erweitern und sich so einem breiteren Publikum zu öffnen? Sind denn SWAP und Co. Ein Symbol von Hightech? In meinen Augen nein. Sie sind ein Übel, das man in Kauf nehmen mußte um in einer höheren Sprache als Assembler (vor 20 Jahren!) zu programmieren und ohne die Hardwarenähe aufzugeben. Heute ist das doch eher anachronistisch. Ich will damit keine Werbung für mein REORDER machen, denn dieses war von Anfang an nur als Quickfix konzipiert. Ich habe mir das nur ausgedacht nachdem ich bei der Implementierung einer 64Bit Arithmetik auf dem Z80 einfach kläglich auf die Schnauze gefallen bin.
Bis ich da etwas am Ticken hatte dauerte es ewig und was lief war so schlapp, daß es unbrauchbar war. Eigentlich wollte ich den Stack komplett verstecken. Aber damals mußte das Projekt fertig werden und danach habe ich mich auch nicht mehr darum gekümmert weil ich eine ganze Zeit lang nichts mehr mit Forth zu tun hatte. Aber eines ist sicher: Das hätte mit Forth so wenig oder so viel zu tun wie mit C oder Basic. Das wäre eine Syntax geworden, die nicht mehr oder nur sehr wenig an Forth erinnert. Warum? Weil der Stack unsichtbar ist. Benamte Parameter und lokale Variablen, Typenprüfung und so (was eben die anderen haben) ist wohl nicht mehr Forth genug. Müßte ich damit auch nur einen Vorzug von Forth aufgeben? Nein, keinen einzigen. Die darunter liegende Maschine ist identisch zu der in Forth, nur die Syntax würde sich ändern.
Na ja, was soll‘s! Mir egal, wenn jemand glaubt, daß man ohne SWAP und Co nicht effizient programmieren kann. Dem wird das Leben schon diesen Zahn ziehen.
--- Ulrich Paul
mailto:upaul@paul.de
http://www.paul.de

Betreff: Raetsel
Martin Bitter hat als einziger versucht, das Rätsel aus Heft 1/2001 zu lösen. Er teilt mir sehr richtig "Hastings 1066" mit. Da das aber in dieser Form, nackig hingeschrieben, mehr nach einem wirklichen Herumraten denn nach einer Lösung aussieht, möchte ich mit der Auflösung noch bis zum nächsten Heft warten. Nehmt bitte die Zahl 1066 einfach als Hilfestellung für weitere Versuche und schlagt (das ist der Sinn des Rätsels) in der Literatur nach. Für Forth könnte das ein Anlaß werden, zwei Festkommapakete in ungewöhnlicher Zahlendarstellung zu entwickeln.
Nur so zur Übung und zum Spaß, ohne die Nützlichkeitsfrage in den Vordergrund stellen zu wollen.

Beh.
Nun ja ... ... aber leider reicht der Platz (hier am Rand) nicht aus, es niederzuschreiben ;-) M.B.

Leserbrief zu REORDER
Von: Ulrich Paul <upaul@paul.de>
Hi Martin,
wie ich schon in meinem letzten Leserbrief angedeutet habe, finde ich die, in Forth implementierten, Stackoperationen einfach unzulänglich. Sie sind es nicht nur im Hinblick auf den Funktionsumfang, sondern vor allem bezüglich der Fehleranfälligkeit. Wer einmal probiert hat, mehr als ein halbes Dutzend Werte auf dem Stack mit ROT, PICK, STUCK (oder war es SUCK?) zu verwalten, der weiß was ich meine oder er ist ein Masochist. Vor ca. 10 Jahren hatte ich ein solches Problem und damals war es für mich einfacher, ein neues Wort in Forth einzubauen als mich mit den primitiven Stackoperationen herumzuschlagen.
Heraus gekommen ist REORDER. Das habe ich in mein FPC eingebaut und seit damals kein einziges Mal mehr SWAP oder DUP oder so verwendet. Es ist state-smart, d.h. Man kann es im Interpreter und im Compiler verwenden.
Was macht das REORDER? Es sorgt dafür, daß der Stack so aussieht wie ich es hingeschrieben habe! Ein Beispiel: REORDER (a b c d – a c b a d c c b a) Nach der Ausführung liegen die Werte tatsächlich so auf dem Stack wie es auf der rechten Seite des „Stackkommentars“ (der hier KEIN Kommentar ist, sondern syntaktischer Bestandteil von REORDER!) steht.
Es gibt keine Einschränkung bezüglich der Komplexität oder der Namen im „Stackkommentar“. Allerdings gibt es syntaktische Einschränkungen:

1.) Auf der rechten Seite dürfen nur Namen vorkommen, die auf der linken vorkommen.

2.) Die Namen auf der linken Seite müssen alle verschieden sein.

3.) Alle Namen haben nur eine Lebensdauer bis zur schließenden Klammer. Es wird nichts in das Dictionary eingetragen und daher sind diese Namen außerhalb des betreffenden REORDERS nicht definiert.

4.) Aus Punkt 3 folgt, daß man ohne weiteres Namen wählen kann, die existierenden Worten entsprechen. Es gibt keine Kollision, da beide in unterschiedlichen Namensräumen liegen.

Zum Beispiel verstößt dieser Code gegen diese beiden Regeln: REORDER (a a b c – a b d)
Es kommt „a“ auf der linken Seite zweimal vor und „d“ ist auf der linken nicht definiert worden.
Ich habe damals dem Tom Zimmer den Sourcecode geschickt, aber nie eine Antwort erhalten. Das ging ihm wahrscheinlich doch zu sehr gegen den Strich. Man stelle sich vor: Ein Forth ohne SWAP und so. Man schreibt hin, wie der Stack aussehen soll und den Rest macht der Computer. Das kann doch jeder! Wo bleibt denn da die Kunst? Und so kümmert mein REORDER in meiner Werkzeugkiste still vor sich hin.
Die Geschwindigkeit von dem? Auf dem FPC ist nur ein DUP und ein SWAP etwas schneller (um so 10%). Das OVER ist es nicht mehr. Alles andere wird haltlos in den Schatten gestellt, vor allem wenn mehr als eine dieser Operationen in einer Reihe kommen. Ein SWAP DUP ist auch schon langsamer.

Im Interpreter ist die Geschwindigkeit sowieso kein Problem. Beim Kompilieren nur insofern, daß es halbwegs schnell gehen soll. Aber beim fertigen Code kommt es auf Tempo an. Und deshalb generiert REORDER nativen Maschinencode. Und zwar einen beinahe optimalen. (Man könnte ihn dadurch optimieren, daß er mehr als 2 Register der CPU verwendet. Aber der Aufwand war mir es nicht wert.) Der Sourcecode von REORDER ist in FPC geschrieben, komplett. Für meinen Target-Compiler habe ich das auch für den Z80 portiert. Das ging innerhalb von einem Nachmittag, da alle code-generierenden Worte DEFERs sind. Einfach die neuen Routinen einklinken und ab die Post.
Aber was rede ich? Forth ohne Stackarbeit ist kein Forth und damit ist REORDER ein Sakrileg. So einfach läßt sich die Reaktion, auch die der Forth-Gesellschaft damals, erklären. Denn damals habe ich schon einmal so einen Leserbrief geschrieben. Der ist abgedruckt und ignoriert worden. Jedenfalls habe ich von niemandem gehört, daß er das verwendet. Wenn einer aber den Mut hat, es einmal auszuprobieren, dann bitte: Unter
http://www.paul.de/downloadables/index.htm gibt es den Sourcecode im File reorder.seq. In dem Verzeichnis gibt es noch ein bißchen mehr. Unter dem Link findet man auch eine Beschreibung wie das REORDER intern funktioniert.
CU Uli
PS: Den „Sündern“ und solchen, die es werden wollen, empfehle ich auch das File switch.seq. Das ist auch so ein syntaktischer Zucker und wenig forth-ig. Dafür praktisch. Und wenn ich wieder einmal etwas Zeit habe, dann räume ich den Code für State-Machines auf und lade den Source-Code auch hoch. Was da ungefähr zu erwarten ist, steht auf der Homepage unter „State Machine Preprocessor for C“ im o.g. Verzeichnis. Nur dann eben nicht für C, sondern für Forth. (Die Forth-Version ist viel älter, aber da ich keine Resonanz erwartet habe, ist der Code auch (noch) nicht verfügbar.)
--- Ulrich Paul
mailto:upaul@paul.de
http://www.paul.de

Nachruf

Aus der c‘t 6/2001:
Der Mathematiker Claude Elwood Shannon begründete die moderne Informationstheorie und führte das „Bit“ als Maß der Information ein. Am 25. Februar ist Shannon im Alter von 84 Jahren gestorben.
Der Nachrufartikel aus der c‘t bezieht sich auf das bekannte Buch, das die Informationstheorie begründet, und zählt die Verdienste des verstorbenen MIT-Professors C.E. Shannons aus dieser Forschungsrichtung auf.
Ergänzend zum c‘t-Artikel sollte man vielleicht noch erwähnen, daß auch die uns allen sehr geläufige Schaltalgebra (Anwendung des Aussagenkalküls auf Reihenparallelschaltungen von elektronischen Schaltern) mit dem Namen Shannon verknüpft ist (Trans. A.I.E.E. 57, 1938 und Bell Syst. Techn. J. 28, 1949) und daß Shannon sich als Mitbegründer der Zuverlässigkeitstheorie einen Namen gemacht hat (J. Franklin Inst. 262, 1956). Das von Shannon gemeinsam mit Moore ursprünglich ins Auge gefaßte höchst praktische mathematische Problem war dabei der Aufbau hochzuverlässiger Systeme (Telefonnetze) aus redundant verwendeten minderzuverlässigen Komponenten (Relais). (Man könnte auf solche Weise z.B. Eine Diodendekodiermatrix mit ganz minderwertigen Dioden aus der Wühlkiste zu einem höchst zuverlässigen Tastaturdekodierinstrument zusammenfügen, wenn man es geschickt anfängt und Redundanzen geeignet einbaut.) Zur Berücksichtigung zweier sich einander ausschließender Ausfallarten (Kurzschluß, Leerlauf, Intaktsein) benötigt man keine dreiwertige Logik – die übliche zweiwertige reicht. All das hat Shannon uns vorgezeichnet. Die Treppenmethode zur beliebigen Steigerung der Systemzuverlässigkeit und die S-Förmigkeit mit dem „Knackpunkt“, unterhalb dessen das System schlagartig in sich zusammenbricht, sind keine Selbstverständlichkeiten und deren Einführung, Entdeckung und Erstbehandlung verdienen höchste Anerkennung.
Irgendwie scheinen diese Ideen jedoch in der Luft gelegen zu haben. J. Von Neumann (Ann. Math. Studies 34, 1956): Wir Bioorganismen arbeiten mit redundanten minderzuverlässigen Organen höchst zuverlässig, brechen aber unterhalb des Umkehrpunktes (zu kleine Komponentenzuverlässigkeiten) kollapsartig zusammen. Shannon konnte die Arbeit „Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components“ von J. Von Neumann nicht kennen, da sie im selben Jahr (1956) wie seine eigene erschien. J. Von Neumann erklärt aber ausdrücklich, daß seine Arbeit Aufzeichnungen von Vorlesungen darstellt, die er, J. Von Neumann, im Jahre 1952 gehalten hatte. Er zitiert die Arbeiten Shannons zur thermodynamischen Informationstheorie (Bell Syst. Techn. J. 27, 1948), auf die Relais-Arbeit nimmt er aber natürlich keinen Bezug – kann er ja, so nimmt man an, auch gar nicht, da erst 1956 erschienen. Andererseits betont J. Von Neumann im Vorwort seiner Arbeit aus dem Jahre 1956, daß seine „Neuronen“ (als „Basisorgane“ in einem Automaten) mit gleichem Recht auch hätten „Relays“ genannt werden können. Für den Betrachter ist es sehr interessant zu erfahren, wie stark die Verflechtungen in den Wissenschaften sind und wie schwer es sein kann, Prioritätsfragen sauber zu klären. Shannon arbeitet mit Disjunktion, Konjunktion und Negation (zum Aufbau von Reihenparallelschaltungen), J. Von Neumann bezieht sich auf allgemeine Automaten und arbeitet mit Multiplexing, Mehrheitselementen und Sheffer-Strokes (NAND), also mit jenen Begriffen, die bei neuronalen Vorgängen (in seiner Arbeit als mögliches Anwendungsbeispiel erwähnt) prävalent sind. Die S-Förmigkeit der Zuverlässigkeitsfunktion wird von beiden untersucht. Die Betrachtung zweier sich einander ausschließender Ausfallsarten (Kurzschluß – Relaiskontakte durch Überlastung miteinander verschweißt, Leerlauf – es hat sich ein Verbrennungsrückstand zwischen die Kontakte geschoben) kommt nur bei Shannon (and Moore) vor.

Fred Behringer

Dr. C.H. Ting hat eine Sammlung seiner Arbeiten mit und an Forth in den letzten 20 Jahren auf eine CD gepackt. Sie enthält um 20 Implementationen von eForth, Handbücher, Kurse..usw. Die CD soll 25$ kosten.
http://www.ultratechnology.com/offete.html

Ein einfacher POSTSCRIPT-Druckertreiber für bigFORTH

Bernd Beuster <bernd.beuster@epost.de>

16. Januar 2001

Drucken von Blockdateien unter Linux

In bigFORTH sind große Teile des Betriebssystems im Blockformat geschrieben. Während Dateien im Streamformat problemlos mit herkömmlichen Texteditoren bearbeitet werden können, muß im allgemeinen für Dateien im Blockformat der interne Forth-Editor benutzt werden.

Das Ausdrucken von Streamdateien ist unter Linux mit Programmen wie a2ps (ASCII-POSTSCRIPT-Konverter) komfortabel möglich, bei den Blockdateien ist etwas Mehrarbeit angesagt.

In diesem Artikel soll ein Konverter von Blockdatei nach POSTSCRIPT beschrieben werden. Der POSTSCRIPT-Output kann direkt auf einen Drucker oder in eine Datei umgeleitet werden.

```
(alles in einer Zeile! der setzä)
$ bigforth scr2ps.fs -e 'scr2ps forth.fb bye' | lpr
(alles in einer Zeile! der setzä)
$ bigforth scr2ps.fs -e 'scr2ps forth.fb bye' >
 forth.ps
```

Eine kurze Einführung in POSTSCRIPT

POSTSCRIPT wurde als universelle Druckerbeschreibungssprache von der Firma Adobe in den 80er Jahren entwickelt. Der POSTSCRIPT-Interpreter ist ähnlich dem Forth Interpreter. Ein Programm besteht aus Definitionen von Variablen, Prozeduren, Directories und aus einem interpretativen Teil, welcher den eigentlichen Druckjob darstellt.

Der Datenstack von POSTSCRIPT ist im Gegensatz zu Forth objektorientiert, z.B. ist der Operator `add` sowohl für Integer- als auch für Fließkommazahlen gültig.

Ein "Hello World!" sieht in POSTSCRIPT folgendermaßen aus:

```
%!PS-Adobe-1.0
/TimesNewRoman findfont 12 scalefont setfont
72 842 144 sub moveto
(Hello World!) show
showpage
```

Kommentare werden in POSTSCRIPTmit einem % eingeleitet, der Rest der Zeile wird ignoriert. In Unix dienen die ersten 16 Bit einer Datei als „Magic Number“, um deren Dateityp zu identifizieren. Der String `PS-Adobe-1.0` hilft den POSTSCRIPT-Interpretern, anhand der Versionsnummer die Interpretation zu steuern.

Nun wird der Font eingestellt. Der Slash sorgt bei `/Times-NewRoman` dafür, das dieser String als Literal verwendet werden soll und nicht etwa interpretiert wird.

findfont (key — font) liefert das Fontdirectory

scalefont (font scale — font) liefert ein neu skaliertes Fontdirectory

setfont (font —) setzt den aktuellen Font

Die Größe des Fonts soll 12 pt betragen, 1 pt sind defaultmäßig 1/72 Zoll. Nach dem der Font selektiert wurde, wird die Zeichenposition eingestellt.

moveto (x y —) setzt die aktuelle Zeichenposition auf die User-Space-Koordinaten (x,y)

Eine DIN-A4 Seite hat die Abmaße 595 pt mal 842 pt. Der Koordinatenursprung liegt in der linken, unteren Ecke. Daher wird der auszugebende String 1 Zoll vom linken Blattrand und 2 Zoll vom oberen Blattrand positioniert.

show (string —) schreibt den String an der aktuellen Zeichenposition

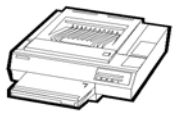
Strings werden in POSTSCRIPT mit () begrenzt. Um auch die Klammern selber auszugeben, muß man ihnen jeweils einen Backslash voranstellen.

Bis zu diesem Zeitpunkt wurde noch kein Text ausgegeben, sondern nur die Seite im internen Speicher des Druckers aufbereitet. Die eigentliche Ausgabe der Seite geschieht mit `showpage`.

Nun wollen wir ein Programm schreiben, daß mehrere Zeilen untereinander ausgeben kann.

```
%!PS-Adobe-1.0
% move to the next line
/n { % ( -- )
 /y0 y0 bfs sub store
 x0 y0 moveto
} bind def

% show and move to the next line
/N { % ( string -)
 show n
} bind def

% init variables
/sw 842 def % upper boarder
/bfs 11 def % font scale
/x0 72 def % initial x position of line
/y0 sw 144 sub def % initial y position of line

% output lines
/Courier findfont bfs scalefont setfont
x0 y0 moveto (The dark brown fox) N
(jumps over) N
n
(the lazy dog.) N
showpage
```

Variablen und Prozeduren werden mit dem `def`-Operator definiert.



def (key value —) Assoziiert *key* mit *value* im aktuellen Directory

Die Variable `sw` hat somit den Integerwert 842, wogegen die Variable `y0` den Integerwert 698 zugewiesen bekommt.

Prozeduren werden in einer Liste, welche mit {} begrenzt ist, definiert und ebenfalls einem Key zugewiesen. Der Operator `bind` dient zur schnelleren Abarbeitung der Prozedur während der Programmabarbeitung.

Die Prozedur `n` dekrementiert also die Variable `y0` und positioniert somit die aktuelle Zeichenposition an den Anfang der nächsten, darunterliegenden Zeile. Die Prozedur `N` zeigt den String an und positioniert auf die nächste Zeile.

Das ist fast alles, was wir wissen müssen, um unsere Applikation zu schreiben.

Applikation

Ein neues Vokabular namens PostScript enthält die meisten neuen Worte. Es sind teilweise Redefinitionen von Standardworten, nur daß diese die Strings nicht auf der Standardausgabe ausgeben, sondern an den String im Puffer PAD anhängen. Dieser String wird dann als PostScript-String formatiert auf der Standardausgabe ausgegeben.

Es werden mit TRIAD immer 3 Blöcke auf eine DIN-A4 Seite gedruckt, am oberen Rand der Seite steht der Dateiname.

In Gforth müssen folgende Worte zusätzlich programmiert werden:

CAPACITY (— u) liefert die maximale Anzahl der Blocks

file? (—) schreibt den aktuellen Dateinamen

Literatur

[1] PostScript language tutorial and cookbook,
Addison-Wesley Publishing Company,
ISBN 0-201-10179-3, 1985.

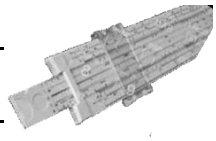
[2] PostScript language reference manual,
Addison-Wesley Publishing Company,
ISBN 0-201-10174-2, ISBN 0201-10169-6, 1985.

[3] PostScript&GHOSTSCRIPT Resources,
http://www.geocities.com/SiliconValley/5682/PostScript.html

```
\ Print screen files
\
\ $Id: scr2ps.fs,v 1.2 2001/01/14 22:37:30 bernd Exp $

FORTH DEFINITIONS
DECIMAL


VOCABULARY POSTSCRIPT


\ increment byte in memory
: C+!  ( n c-addr -- )  DUP C@  ROT +  SWAP C! ;


POSTSCRIPT ALSO  DEFINITIONS

\ get # of pages in SCR file
: #PAGES  ( -- n )  CAPACITY 3 + 3 / ;

: PS-HEADER  ( -- ) ." %!PS-Adobe-1.0"
  CR ." %%Title: " file?
  CR ." %%Creator: scr2ps"
  CR ." %%Pages: " #PAGES U.
  CR ." %%DocumentMedia: a4 595 842 0 () ()"
  CR ." %%DocumentNeededResources: font Courier"
  CR ." %%EndComments"  ;

: PS-PROLOG  ( -- )
  CR ." %%BeginProlog"
  \ move to the next line
  CR ." /n {"
  CR ."   /y0 y0 bfs sub store"
  CR ."   x0 y0 moveto"
  CR ." } bind def"
  \ show and move to the next line
  CR ." /N {"
  CR ."   show n"
  CR ." } bind def"
  CR ." %%EndProlog" ;


\ Initialize page description variables
: INIT-VARIABLES  ( -- )
```


```
  CR ." /sw 842 def"          \ upper boarder
  CR ." /bfs 11 def"          \ font scale
  CR ." /x0 72 def"           \ initial x position of line
  CR ." /Courier findfont bfs scalefont setfont"
  ;

\ append string to PAD
: APPEND  ( c-addr1 u -- ) TUCK  PAD COUNT +  SWAP MOVE  PAD C+! ;


\ Redefine some standard words.
\ Output is written to buffer at PAD.
: SPACE  ( -- )  S"  " APPEND ;
: SPACES  ( u -- )  DUP 0> IF  1- FOR SPACE NEXT  EXIT THEN  DROP ;

: U.  ( u -- )  0 <# #S #>  APPEND ;
: U.R  ( u1 u2 -- )  >R  0 <# #S #>  R> OVER - SPACES  APPEND ;


\ escape special characters
: \EMIT  ( c -- )
  DUP [CHAR] ( =  OVER [CHAR] ) = OR  OVER [CHAR] \ = OR
  IF [CHAR] \ EMIT THEN  EMIT ;

: \TYPE  ( c-addr u -- )  ?DUP 0= IF DROP EXIT THEN
  1- FOR  COUNT \EMIT  NEXT  DROP ;


\ Type string in PAD enclosed with paranthesis and clear buffer.
: (TYPE  ( -- )   ." (" PAD COUNT  \TYPE  ." )"  0 PAD C! ;

\ type screen number
: .SCR#  ( -- )  S" SCR# " PAD PLACE  SCR @ U. ;

\ type line u in block
: .LINE ( n -- )   DUP 2 U.R  SPACE
  SCR @ BLOCK  SWAP C/L * +  C/L -TRAILING APPEND ;

\ type screen with PostScript commands
: LIST  ( u -- )  SCR !   CR .SCR# (TYPE ." N"
  0  L/S 1- FOR  CR DUP .LINE (TYPE ." N"  1+  NEXT  DROP ;

: NEWLINE  ( -- )  CR ." n n" ;

\ One page contains three screens
: TRIAD  ( n -- )  3 / 3 *
  CR ." %%Page: ? ?"          \ help Ghostview
  CR ." x0 sw 72 sub moveto"  \ goto title string position
  CR ." (" file? ." ) show"   \ bigFORTH: show file name at top
  CR ." /y0 sw 144 sub def"   \ initial y position of line
  CR ." x0 y0 moveto"         \ goto origin
  2 FOR  DUP CAPACITY U<  IF DUP LIST NEWLINE  1+ THEN  NEXT DROP
  CR ." showpage" ;

\ type complete file
: LISTALL  ( -- )
  0  CAPACITY 3 /  FOR  DUP TRIAD  3 +  NEXT  DROP ;

: PS-MAIN  ( -- )  INIT-VARIABLES LISTALL CR ;


FORTH DEFINITIONS

: SCR2PS  ( filename ( -- ) USE  PS-HEADER PS-PROLOG PS-MAIN ;


\\ Usage:
bigforth scr2ps.fs -e "scr2ps minos.fb bye" > minos.ps
```

Ein bedeutender Softwareimperialist bekommt eine transzendentale Anwandlung: Er will es wissen! Jetzt! Sofort! Endgültig!
Er startet seinen besten Rechner und tippt ein: „Gibt es einen Gott?“ Der Rechner: „Not enough memory!“ Also alle Firmenrechner des Imperiums vernetzt und noch einmal:“ Gibt es einen Gott?“ -- „Not enough memory!“ Alle Mitbewerber angerufen, weltweit alle Firmennetzwerke verknüpft: „GIBT ES EINEN GOTT?!“ --- Antwort: „ Billy, - Jetzt ja!“



Viele unserer Leser kennen die Zeitschrift „Spektrum der Wissenschaft“ die deutsche Version der „Scientific American“. Vor einiger Zeit erschien dort ein Artikel über das Rechnen mit langen Zahlen, der meiner Meinung nach (M.B.) jedem Forthprogrammierer aus der Seele spricht. Wer empfindet nicht ein leichtes Unbehagen, wenn er um den Preis der Genauigkeit mit floting point in großen Zahlenbereichen rechnet? Sind Forthprogrammierer nicht geborene 'Integerrechner'? Bedeutet integer nicht unversehrt, unbeeinträchtigt, über allen Zweifel erhaben? Das sind Fragen die in dem folgenden Artikel angesprochen sind. Die Redaktion möchte das Vergnügen diesen Artikel zu lesen mit den Mitgliedern teilen. Herr Pöppe, Redakteur und Autor bei „Spektrum der Wissenschaft“ hat freundlicherweise diesen Nachdruck gestattet.

Rechnen mit garantierter Genauigkeit

Die in die Computer eingebauten Rechenverfahren leiden unter Problemen, die im Extremfall die Ergebnisse völlig unbrauchbar machen können. Dies muss man beim heutigen Stand der Technik nicht mehr in Kauf nehmen. Fortgeschrittene Verfahren liefern Rechenergebnisse mit der Sicherheit eines mathematischen Beweises.

Von Christoph Pöppe

***Aus Spektrum der Wissenschaft September 2000**, Seite 54*

Die Redaktion der Vierten Dimension hat die Rechte diesen Artikel nachzudrucken nur zur einmaligen Verwendung bekommen. Deshalb können wir diesen Artikel hier nicht nocheinmal wiedergeben. Aber Sie können unter der Internetadresse:

http://www.wissenschaft-online.de/abo/spektrum/archiv/5116

diesen Artikel 'downloaden'.

Bei Erscheinen der Vierten Dimension war das noch ein kostenloser Service. Leider hat sich dies inzwischen geändert.

Bei der Erstellung der PDF-Version (2002) kostete dieser 'Download' 1,25 €.

Alle Kombinationen von k Einsen in einem n-Bit-Wort in High-level-Forth

Fred Behringer <behringe@mathematik.tu-muenchen.de>

Wil Baden hat kürzlich in der Forthwrite 110 einen Artikel über die Bildschirmdarstellung aller Permutationen von n Objekten veröffentlicht. Chris Jakeman hat im selben Heft auf ein Programm von Ed Hersom aus dem Jahre 1991 aufmerksam gemacht, das kürzer mit **RECURSE** arbeitet. Das gab mir die Anregung, einen Gedanken aus der Zuverlässigkeitstheorie in Forth (Turbo-Forth) wiederaufzugreifen. Zuverlässigkeit ist Wahrscheinlichkeit des Intaktseins. Es geht dabei um n-komponentige Systeme, die selbst und deren Komponenten zwei Zustände annehmen können, intakt sein oder nicht intakt sein. Zur Beschreibung verwendet man boolesche Funktionen von n Variablen. Solche Funktionen lassen sich als Disjunktive Normalform (DNF) darstellen. Arithmetisiert (0 und 1 als Wahrheitswerte, +, * und - als Operationen), lassen sich boolesche Funktionen auch als Multilinearform (MLF) darstellen. Beides hat Vor- und Nachteile. Leichter zugänglich sind DNFs.

Es gibt viele andere Anwendungsgebiete für boolesche Funktionen. Man denke im Sinne unserer aktuellen Diskussionen an Ralph Hempels Lego-Roboter, die um sagen wir 16 selbstgebaute Sensoren erweitert wurden. Zur Verarbeitung der Kombinationen von (boolesch interpretierten) Eingangssignalen wird man eine Funktionstafel verwenden und die DNF aufstellen.

Bei der Umwandlung einer DNF in die MLF braucht man ein systematisches Verfahren zur Bestimmung der Koeffizienten der MLF. (Die Koeffizienten können beliebig ganzzahlig sein, positiv oder negativ.) Man kann das Problem stufenweise angehen. Auf Stufe k benötigt man sämtliche Kombinationen von k Einsbelegungen in einem gegebenen n-Tupel von booleschen Variablen. Das läuft auf die Bestimmung aller Kombinationen von k Einsen in einem "Wort" der Länge n hinaus. (Im untenstehenden Programm werden bis zu 16 Bits an Wortlänge zur Verfügung gestellt. Für n kleiner als 16 bleiben einfach führende Nullen stehen.)

Die Einsen werden in meinem Programm über Zweierpotenzen zusammengestückelt (High-level-Forth ohne Effizienzbetrachtung). Für die Ablage der verwendeten Parameter-Tripel habe ich nicht den Returnstack vorgesehen. Alles wird auf dem Datenstack abgelegt, da an den Tripeln, auch übergreifend, viele Zwischenoperationen durchzuführen sind, für die der Returnstack zu umständlich wäre. Ich verwende viele **PICK**s. Das ist unschön. Vielleicht hätte sich das vermeiden lassen, wenn ich wenigsten den "jüngsten" Parameter des jeweilige Tripels auf den Returnstack gelegt hätte. Da sehe ich Variationsmöglichkeiten.

Algorithmus: Man fülle das "Wort" der Länge n mit der verlangten Anzahl von Einsen, sagen wir k, von rechts nach links auf.

Ausgehend von irgend einer schon erreichten Kombination von k Einsen im n-Bit-Wort, sagen wir 11000101, verschiebe man die von links her gesehen erste von einer 0 angeführte 1 um ein Bit nach links, hier also 11001001. Dann starte man den Prozeß mit dem kleineren Problem, das man erhält, wenn man den "rechten" Teil des Wortes, hier 1001, beibehält, was in unserem Beispiel heißt, man nehme 00001001 als neuen Ausgangspunkt und vervollständige den "linken" Teil des Wortes, hier also 0000, auf jede mögliche Art um zwei weitere Einsen.

Erläuterung: Die mit dem untenstehenden Programm erhaltene Folge von jeweils drei Einserkombinationen in sechs Bits, die von links nach rechts, Zeile für Zeile, zu lesen ist, soll die Wirkungsweise des Algorithmus erläutern.

Die zweite, dritte und vierte Bitgruppe haben jeweils **011** als festgehaltenen rechten Teil, und das kleinere Problem besteht darin, alle "Kombinationen" einer einzigen Eins in drei Bits, dem verbleibenden linken Teil, zu finden. In den Bitgruppen 5 bis 7 wird der rechte Teil der Bitgruppe 2 genommen, die linke Eins um ein Bit verschoben und das Restproblem, jeweils immer eine Eins links anzufügen, mit dem nunmehr "festgehaltenen" Teil 000101 angegangen.

Implementation: Der **ELSE**-Teil des ersten **RECURSE** erzeugt k Einsen. Als Kontrollmechanismus wirkt der auf dem Datenstack liegende Parameter "count". Das Verschieben der mit einer führenden 0 versehenen

000111 001011 010011 100011 001101 010101 100101 011001 101001 110001

001110 010110 100110 011010 101010 110010 011100 101100 110100 111000

weitestlinken 1 um eine Bitstelle nach links wird vom **IF**-Teil des ersten **RECURSE** übernommen. Die Rückwärtsrekursion, also der Wiederaussprung aus einer Rekursionsstufe, für den Fall, daß eine 1 links aus dem "Wort" herausfällt, ist Sache des zweiten **RECURSE**. An sich ist das im negativen Sinne zu verstehen: Das zweite **RECURSE** nimmt die (Vorwärts-) Rekursion wieder auf. Zuvor wird, falls eine 1 links herausgefallen ist, unabhängig davon, wieviele Einsen vom "count" auf dem Stack schon "gesammelt" wurden, der Restteil des Wortes (ohne die herausgefallene 1) in die Variable **ACCU** gelegt. Dann wird **RECURSE** solange übersprungen, also die Rückwärtsrekursion solange aufrechterhalten, bis der Inhalt von **ACCU** mit dem irgendwann zwangsweise auf dem Stack anzutreffenden Parameter "accu" übereinstimmt. Erst dann wird die Vorwärtsrekursion wiederaufgenommen, und zwar diesmal eben über das zweite **RECURSE**. Der gesamte Prozeß kommt zu einem Ende, wenn man wieder bei den anfänglichen Parametern (level,count,accu = 0,0,0) angelangt ist.

Bemerkung: Wie leicht einzusehen, kann die Rekursionstiefe nie die Bitzahl im "Wort" übersteigen. Mit der Rückwärtsrekursion gelangt man an die Parameter "level" und "count" die der in ACCU festgehaltenen "Knotenkombination" (dem "root" im "subtree") zugeordnet sind, von welcher aus dann wieder eine Vorwärtsrekursion (mit einem kleineren Problem) aufgenommen werden soll. Beim Hinsehen sieht man sofort, wo die am weitesten links plazierte 1 mit führender 0 steckt. Aber selbst wenn man annimmt, man könne auch das Programm so einrichten, daß es die Suche nach dieser 1 mit führender 0 in einem einzigen Schritt erledigt, hätte man dann wiederum das Problem, daß die Zahl der benötigten Schritte, die ja dann nur Vorwärtsrekursionen wären, unerträglich groß werden könnte (Returnstackbelastung). Im untenstehenden Programm kann das nicht passieren. Wir haben ein ständiges Auf und Ab, Vorwärts- und Rückwärtsrekursion wechseln sich ab (vorwärts im "subtree" des jeweils "kleineren" verbleibenden Problems, dann zurück bis zum letzten "node" und dann entlang eines anderen Zweigs wieder vorwärts).

Zeitbetrachtungen: Es gibt n über k Kombinationsmöglichkeiten von k Einsen aus n Bits. (Aus der Konstruktion des Binomialkoeffizienten n-über-k heraus (Produkt der größeren k natürlichen Zahlen kleiner/gleich n geteilt durch das Produkt der ersten k natürlichen Zahlen) ersieht man, daß bei gegebenem n der größte Binomialkoeffizient für k = [n/2], für die größte ganze Zahl kleiner/gleich n/2, erreicht wird. Die größte vom untenstehenden Programm noch verarbeitete Bitzahl ist n = 16d. Die größte vom Programm zu verarbeitende Zahl von Kombinationen in einer Gruppe ist also 16 über 8, d.h. 12870. Meine Zeitbetrachtung richte ich auf diesen "schlimmsten Fall" aus. (Zur Probe: Die Summe aller Binomialkoeffizienten n-über-k für k = 1 bis n ergibt 2^n.) Auf einer 486/66-Maschine benötigte der erwähnte schlimmste Fall etwa 1 Sekunde bei Verzicht auf die Bildschirmanzeige und 48 Sekunden, wenn man sich die Einserkombinationen der Reihe nach anzeigen läßt.

```
HEX

HERE
  1 ,   2 ,   4 ,   8 ,   10 ,   20 ,   40 ,   80 ,
100 , 200 , 400 , 800 , 1000 , 2000 , 4000 , 8000 , 10000 ,
\ Die 10000 fuer den Fall, dass bei 16 Bit eine 1 links herausgeschoben wird.

: 2-NTH ( n -- 2^n )                \ 0 <= n <= 1F
   [ DUP ] LITERAL                  \ In Turbo-Forth muss die Stackbilanz ge-
   SWAP 2* + @ ; DROP               \ sichert bleiben; daher [ DUP ] und DROP.

VARIABLE #BITS                      \ "Wort"laenge
VARIABLE #ONES                      \ Zahl der Einsen in den Kombinationen
VARIABLE ACCU                       \ Knotenwert fuer Rueckwaertsrekursion

: (K-IN-N) ( level count accu -- )
   2 PICK #BITS @                   \ lv cn ac lv (#b)
   #ONES @ - 3 PICK +               \ lv cn ac lv (#b)-(#o)+cn ; noch Bits
   <= IF                            \ frei ? Sonst zurueck auf Stufe vorher.
     OVER #ONES @ =                 \ Schon genuegend Einsen in count ?
     IF
       BASE @ OVER 2 BASE ! CR 0    \ Naechste Einserkombination am
       <# #BITS @ 0 DO # LOOP       \ Bildschirm anzeigen.
       #> TYPE SPACE BASE !         \ In Basis 2 mit fuehrenden Nullen.
       3DUP 2 PICK 1- 2-NTH -       \ Ersetze 1 ganz links durch 0,
       DUP ACCU !                   \ bewahre "Rumpf"wert in ACCU auf,
       ROT 1+ -ROT 2 PICK 1-        \ und verschiebe besagte 1
       2-NTH +                      \ um eine Bitstelle nach links.
     ELSE
       3DUP ROT 1+ ROT 1+ ROT       \ lv cn ac lv+1 cn+1 ac
       5 PICK 2-NTH +               \ lv cn ac lv+1 cn+1 ac-neu
     THEN
     RECURSE
   THEN
   3DUP + + 0<>                     \ Weiter im "Rueckwaertsgang" ?
   IF
     DUP ACCU @ =                   \ Weiter im "Rueckwaertsgang", falls der
     IF                             \ "Rumpf"wert noch nicht erreicht.
       3DUP 2 PICK 1- 2-NTH -       \ Rekursion wiederaufnehmen, also 1 ganz
       DUP ACCU !                           \ links durch 0 ersetzen, "Rumpf"wert
```

```
      ROT 1+ -ROT 2 PICK 1-          \ in ACCU speichern, besagte 1 um eine
      2-NTH +                        \ Bitstelle nach links verschieben
      RECURSE                        \ und weiter im Rekursionsprozess
    THEN
  THEN
  2DROP DROP ;                       \ 1 Schritt zurueck und Parameter weg.

: K-IN-N ( #ones #bits -- )
  DUP 1      10 BETWEEN NOT          \ Fehler, wenn "Wortlaenge" ausser Bereich
  ABORT" Eingabefehler" #BITS !
  DUP 1 #BITS @ BETWEEN NOT          \ Fehler, wenn Einserzahl ausser Bereich
  ABORT" Eingabefehler" #ONES !
  0. 0 (K-IN-N) ;
```

Ende gut, alles gut!

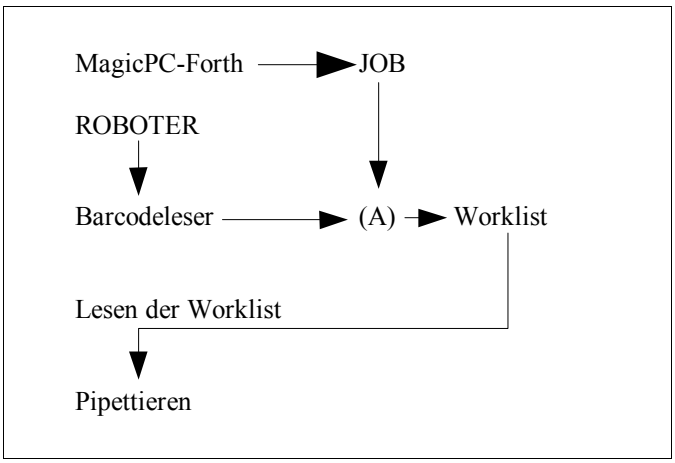
Rainer Saric

Hammerstr. 3b
65594 Runkel
EMail: Rainer.Saric@t-online.de
Tel.: 06482 - 60352

Bemerkung: Aus verständlichen Gründen, kann ich nicht alle Einzelheiten des Projekts schildern!

Problem: Es sollte ein Programm installiert werden, das aus normierten Kunststoffplatten (384 Bohrungen) mittles eines Pipettierroboters Proben löst und Teilmengen aus diesen Platten entnimmt. Die entnommenen Megen sollten protokolliert und bei der grafischen Darstellung der Platten farblich gekennzeichnet werden.

Die Erstellung des Programms durch den Roboterhersteller erschien mir etwas zu teuer (>10 TDM), so da× ich mich kurzerhand an die Realisierung in Forth machte. Da das Steuerprogramm des Roboters unter NT lief, wählte ich für ich als Schnittstelle Win32For. Dieses Programm, nennen wir es A, hatte lediglich den Zweck die über einen Barcodeleser gelesene Platte zu identifizieren und die zur Pipettierung nötige Worklist aus einem JOB-File für den Roboter zu erzeugen.



Für die Realisierung des Win32For-Programms benötigte ich 1/2 Tag, der Quelltext besteht aus ca. 100, unspektakulären Zeilen:

```
~~~
struct{
  16 chars .plate
   4 long  .source
   4 long  .destination
    4 long .total
    4 long .volume
}struct WORKLIST

create worklist{ sizeof WORKLIST MAXVIALS * allot

create aspirate$ 96 allot
~~~
: separator  ( -- c-addr n )   s" ;" ;
: A;         ( -- c-addr n )   s" A;" ; \ Aspirate Befehl
: D;         ( -- c-addr n )   s" D;" ; \ Dispense Befehl
: W;         ( -- c-addr n )   s" W;" ; \ Wasch Befehl
: microplate ( -- c-addr n )   s" Greiner384 HalfWell,
 landscape;" ;
: hitplate     ( -- c-addr n )   s" VialCarrier;" ;
~~~
\ Füllen des Roboters
: fuellen      ( volumen -- c-addr n )
   s" A;System;;;1;;" aspirate$  place
   s>d <# #s #>       aspirate$ +place
   s" ;;;"            aspirate$ +place
   aspirate$ count ;
~~~
: create-worklist ( -- )
   create-worklist-file
   if
    #vials @ 0 do
     i >worklist count barcode count compare 0=
     if
      I >worklist WORKLIST .source @ source-pos !
      I >worklist WORKLIST .volume @ volume !
      I >worklist WORKLIST .total @ syringe ! loesen
      D;
      I >worklist WORKLIST .destination @
      I >worklist WORKLIST .volume @ write-d
      write-w
     then
    loop
   then
   fidWorkList close-file drop ;
~~~
: main   ( -- )
   ?arg 0= if 2drop s" Error! No parms!" then barcode
 place
   read-job               \ file ist bekannt \..\*****.job
   barcode count doing place
   get-plate              \ und verify
   Start: HelloWindow     \ zeige was du tust
   create-worklist
   2000 ms                \ ein bisschen zeit muss sein
   Close: HelloWindow
   bye ;

 ' main turnkey hits
```

Die erzeugte Worklist ist ein simpler ASCII-Text. Zu meiner Überraschung klappte die Einbindung des Programms auf Anhieb.
Mit mehr Aufwand war die Auswahl der Proben aus einer Oracle-Datenbank verbunden.

Zunächst mußten die für das Programm relevanten Daten, durchaus einige Tausend Zeilen, exportiert werden. Da die Auswahl grafisch erfolgen sollte wählte ich zur weiteren Aufbereitung mein Forth für den Atari (teils basierend auf J.Plewe's FForth), denn ich hatte keine Lust mein GEM-Windows-Interface auf NT zu portieren und die Emulation (MagicPC) war schnell genug.. Über das integrierte "make" war es zudem möglich, ein Programmfile zu erzeugen, dass alle nicht benötigten Worte entfernte. (Programmgröße nach dem "strippen" ca. 80kb). Die Dialoge, Fenster, etc wurden über Resource-Constructionset entworfen und über Systemfuktionen eingebunden.

Das Programm erfüllte folgende Aufgaben:

a) Datenbank
b) Konvertierung von CSV-Files
c) Listen aller Proben
d) grafische Darstellung der Ergebnisse
e) grafische Darstellung der Platten und der Auswahl daraus
f) Exportfunktion für das Programm A
g) Drucken der Ergebnisse

Die Realisierung der Datenbank (siehe structs{) war ebenfalls relativ einfach, da durch den Roboter eine klare Struktur vorgegeben war:

Platte -> 8 Töchter -> 8 Enkel -> 44 Urenkel

Erzeugung der Arbeitsliste für (A)

```
: ...
  #vials 0 DO
   I ValidSelected IF ExportVial THEN
  LOOP ;
```

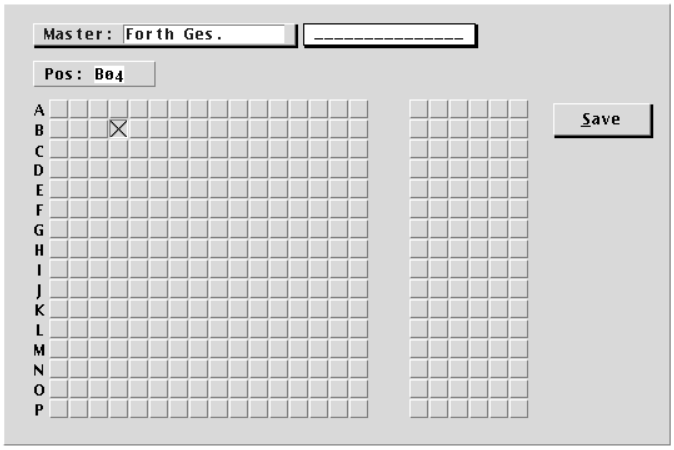
```
\ structs
\ An implementation of simple data structures
\ 1999/2000 Rainer Saric

: chars           ( size -- size' )
  CREATE OVER , + IMMEDIATE
  DOES>   @ STATE @ IF     POSTPONE +  ELSE + THEN ;

: rect            8 chars ;
: lrect           16 chars ;
: long            4 chars ;
: word            2 chars ;
: pointer 4 chars ;
: void            4 chars ;
: byte            1 chars ;

: struct{  ( -- 0 )
  GET-ORDER GET-CURRENT ALSO WORDLIST DUP SET-CURRENT DUP
  CONTEXT ! 0 ;

: }struct ( *list size -- <.....> )
  >R >R SET-CURRENT SET-ORDER R> R>
  CREATE SWAP , , IMMEDIATE
  DOES> @ >R
                  BL WORD COUNT R> SEARCH-WORDLIST 0=
  ABORT" no struct member !!"
                  EXECUTE ;

: sizeof  ( -- <...> ) ' >BODY 4+ @ STATE @ IF [COMPILE]
  LITERAL THEN ; IMMEDIATE
```

Bei der grafischen Darstellung wurden Werte über dem "Wirksamkeitswert" farblich hervorgehoben. Die Grafik passt sich automatisch an min/max an.

```
\ Tabelle WaveLenght

: ,24 ( addr -- )
    24 0 DO
            0 0 BL PARSE >NUMBER
            2DROP DROP
            [COMPILE] ,
          LOOP ;

CREATE     wlLimit                \ UV-Limit überschritten
,24 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
,24 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
,24 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
,24 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
~~~
```

Wichtige Werte konnten über eine Dialogbox gesetzt werden.

Die Auswahl erfolgte per Maus sowohl über das Ergebnis, des Textes oder grafisch.



Zur leichteren Verfügbarkeit wurde die erzeugte Worklist auf ein Netzlaufwerk geschrieben.

Zu meiner Person: 44 Jahre, Chemotechniker, u.a. Forth als Hobby

Leider wurde mein Arbeit nicht hinreichend gewürdigt. Allerdings nahm der Hersteller des Roboters über eine Personalberatung (Ich kam mir richtig wichtig vor) Kontakt mit mir auf.

Seit dem 1.3. bin ich bei dieser Firma tätig.

Bibliothekskonzepte

JavaBeans

Teil I oder II, jenachdem, welchen man zuerst liest

Jörg Staben

(nach: Java für Dummies, MITP Verlag)

Wenn Sie keine eigenen Bohnen anbauen wollen, kaufen Sie doch ein paar JavaBeans

Die Entwickler von Java wollen kommerzielle Softwareentwickler dazu ermutigen, Java-Komponenten zu erzeugen und zu verkaufen.

Diese Java-Komponenten können von den Anwendern kombiniert und benutzt werden, um eigene benutzerdefinierte Applets (=Java-Anwendungen für´s Internet) zu programmieren. Aus diesem Grund haben die Entwickler von Java den **JavaBeans-Standard** geschaffen.

Der JavaBeans-Standard ermöglicht Entwicklern, Java-Klassen (=Java-Programme) zu veröffentlichen, ohne die innersten Geheimnisse des Codes enthüllen zu müssen.

Solche Klassen kann der Anwender modifizieren und anpassen. Der Anwender kann die öffentlichen Methoden und Variablen eines JavaBean-Objekts nachschauen. Zudem haben JavaBeans-Klassen Methoden, die leicht anzupassen sind.

Sie wurden so entworfen, daß sie sich ohne Probleme in die visuellen Bearbeitungswerkzeuge der IDEs einfügen. Der Anwender muß aber keine IDE benutzen, um JavaBeans benutzen können.

Zwar muß der Anwender eines JavaBean etwas Arbeit investieren, um herauszufinden, wie man dieses JavaBean benutzt, aber viel weniger, als z.B. zur Erstellung eines vollständigen Spreadsheets oder Datenbank-Viewers oder was auch sonst immer erforderlich wäre.

Warum sollen nur VB-Programmierer den ganzen Spaß haben?

Der Anwender wird zusehends in Lage sein, den meisten Java-Code wiederzuverwerten, ohne eine einzige Zeile zu tippen. Statt dessen wird einfach das Bild einer JavaBean auf ein Bild eines Java-Programms gezogen und dort abgelegt.

Auf diese Weise recyceln z.B. Visual Basic- und Delphi-Programmierer Code; es ist eine wirklich wirkungsvolle Technik. Und genau darin liegt die Magie der JavaBeans.

Java-Klassen wiederzuverwerten kann eine Qual sein, wenn Sie im Javadoc nachschauen oder den Quellcode lesen müssen, um die Eigenschaften oder Methoden von Klassen herauszukriegen.

Eine JavaBean ist ein Stück Code - eine Komponente die der visuellen Programmierumgebung und anderen Werkzeugen nach einem vereinbarten Standard über seine Methoden und Eigenschaften berichtet. Visuelle Programmierwerkzeuge wie Jbuilder, Symantec Cafe oder Forte4Java können eine JavaBean inspizieren, ihre Methoden und Eigenschaften kennenlernen und diese Informationen dazu verwenden, dem Programmierer das Leben leichter zu machen.

Forth-Quellcode oder JavaKomponente?

Anstatt in einem endlosen Quelltext die richtige Variable zu suchen, um dann im Quellcode von Hand beispielsweise die Hinterundfarbe eines Menüs in „rot“ umzuändern, kann eine visuelle Programmierumgebung für eine JavaKomponente ein Menü von gültigen Farben bereithalten, aus dem man einfach den entsprechenden Wert auswählt.

Eine solches Herangehen an die Wiederverwendung von Code über Komponenten ist komfortabler, vor allem auch sicherer und weniger fehleranfällig als die manuelle Veränderung schlimmstenfalls kaum nachvollziehbarer Quelltexte: Kein Tippfehler wie „rpt“ statt „rot“, kein ungültiger Wert wie „ultra-smaragdgrün“ für die Hintergrundfarbe.

Dahinter steht ein für Forth-ler völlig ungewohnter Gedanke:

Vielleicht will man den Quelltext überhaupt nicht haben, sondern einfach nur die Funktionalität eines gebrauchsfertigen Moduls nutzen...

P.S.:
Nur so erwähnt, ohne dass ich die genaue Bedeutung kenne:

Der JavaBeans-Standard bietet übrigens auch die Fähigkeit der Persistenz; also Daten zu speichern und wiederherzustellen.

Stimmen der anderen: Ist Forth keine Sprache?

Fred Behringer <behringe@mathematik.tu-muenchen.de>

Welch unendlich große Mühe hatte ich im Zelt der Forth-Gesellschaft aufzubringen, den Besuchern der Vaterstettener Media-Night (VD 4/2000) zu erklären, warum ich die Lego-Roboter in Forth programmieren will und woher ich meine Begeisterung für Forth, und zwar ausschließlich für Forth, nicht für das Programmieren an sich, für Forth und nur für Forth, nehme! Der Erfolg hielt sich in Grenzen. Meine Begeisterung konnte ich nicht übertragen. Eine griffige Formel fehlte. Umso gelegener kommt mir eine Meldung von

John Jones aus der comp.lang.forth, die Chris Jakeman ohne zu zögern in der Forthwrite 110 wiedergegeben hat. John Jones faßt zusammen, womit sich sicher viele von uns identifizieren können. Paul Bennett, in der Forthwelt wohlbekannt, zollt höchste Anerkennung. Ich schließe mich an. Hier ist die Übersetzung der Meldung von John Jones:

"..... Mein Hauptanliegen bei der Beschäftigung mit Forth in einem Umfeld der Geschäftswelt besteht darin, Forth als Maschine zu betrachten, nicht als Sprache. Ich gelange immer mehr zu der Überzeugung, rein intuitiv, von der Vernunft her läßt sich das kaum erklären, daß sich die Informatik in einer Sackgasse verrannte, als sie sich dem Sprachenparadigma unterwarf. Damit zwang sie einer einfachen Maschine menschliche Denkstrukturen auf, nur um dann hinterher wieder verzwickte Sprachkonstruktionen zu bemühen, um die eigentlichen Ziele zu erreichen. Das aber trieb Mensch und Maschine auseinander, sehr zum Nachteil des Menschen. In Forth bleibt die Mensch-Maschine-Beziehung transparent. Und trotzdem wird ein gewisses Abstraktionsniveau erreicht, welches hier aber den physikalischen Verhältnissen entspricht. Die "virtuelle Forthmaschine" wird auf die tatsächlich vorhandene Maschine abgebildet. In anderen "Sprachen", und das, meine ich, ist wesentlich, bleibt eine solche Abbildung verborgen oder existiert überhaupt nicht."

Und hier die Übersetzung des Kommentars von Paul Bennett: "Donnerwetter! Das muß man sich merken! Gute Arbeit, John! Da hast Du eine gute Kurzformulierung von Gefühlen und Gedanken herausgearbeitet, die ich auch hatte und habe. Deine Worte beleuchten den wesentlichen Unterschied zwischen Forth und den anderen Sprachen."

Soweit Paul Bennett. Bahnt sich da, so erlaube ich mir, in aller Bescheidenheit zu fragen, etwa ein neuer Paradigmenwechsel an? Weg vom Prokrustesbett der sogenannten gängigen Sprachen, hin zu forthähnlichen offenen Sprachen? Und wenn ja, was heißt dann forthähnlich? Ist Forth überhaupt eine Sprache? Bei John Jones nicht unbedingt. Und wenn schon Sprache, dann doch wohl gleich eine ganze Klasse von Sprachen? Ein Oberbegriff, eine Kategorie für sich. Alles Forthähnliche ist in Forth doch bereits vorgefertigt? Nicht nur die Wortbildung ist unbeschränkt, die Strukturen selbst sind nahezu unbeschränkt erweiterbar. Forth gegen den Rest der Sprachen?

Die Formulierung von John Jones ist gut, ist aber sein Gedanke neu? Die Frage, ob wir wirklich in Sprachen denken, Englisch oder Deutsch, ist schon älter. Mit der möglichen Antwort der Sprachpsychologen, daß die Sprachen uns zu klischeehaftem Denken verleiten. Daß wirklich Neues nur der erfinden kann, der sich außerhalb einer jeden Sprache stellt. Äh, Äh als Denkmittel? Wie oft sitze ich vor einem komplizierten Zusammenhang, möchte mir in immer neuem Anlauf - Genie bin ich nicht - den Durchblick erzwingen und lasse die Synapsen Äh, Äh funken! Brauchen wir zum Denken von Gedanken wirklich eine Sprache?

Zur Kommunikation allerdings brauchen wir eine Sprache. In Forth ist das aber ein dehnbarer Begriff, ein fuzzy set, eine unscharfe Menge. Ein harter Kern und ein fließender Übergang zu den Grenzen hin. Deshalb diskutieren und ringen wir ja auch so sehr um jedes einzelne Wort - Wort im Sinne eines Forthbefehls - im harten Kern. Die Grenzübergänge gestalten wir uns von Fall zu Fall selber, jeder einzelne von uns. Forth als Mittel zur Selbstverwirklichung, zur freien Gestaltung? Können die "anderen" das verstehen? Sollen sie das überhaupt verstehen?

Gehaltvolles

Teil 1

zusammengestellt und übertragen von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien
Nr. 110 Januar 2001

Chris Jakeman begrüßt in seinem Editorial Chris Ranklin aus Leeds als neues Mitglied und hebt John Metthews (siehe auch Seite 8) hervor, der ein englisches Unternehmen zum Übergang zu Forth als firmenempfohlene Programmiersprache bewegen konnte.

3 Forth News
Dave Abrahams <d.j.abrahams@cwcom.net>

Umfangreiche Glossare für Pygmy-Forth und Quartus-Forth - ProForth 3.22 (auch Probeversion) - SwiftX 3.0 - Quartus Forth für Palm OS mit MathLib - WearLogic - Java-Quelltext zu DELTA Forth mit Infix-Postfix-Konverter jetzt kostenlos - Ficl 2.04 kostenlos - StrongForth 0.03 kostenlos - KForth für Windows (95/98/NT) kostenlos - GForth 0.5.0 mit überarbeiteter und erweiterter Dokumentation kostenlos - lib4th 0.0.51 als Linux-dll - Camel Forth für Z80, Z180 und Rabbit 2000 kostenlos - CTRAN, ein Formel-Code-Übersetzer - Win32Forth toolset mit über 2000 Fehlermeldungen - neues 4-Autoren-Lego-Roboter-Buch mit zwei Kapiteln von Ralph Hempel über pbForth - Anton Ertls Diskussionsforum für ANS-Verbesserungen - Martin Bitter als Roboter-Preisgewinner

8 Forth Preferred as Development Environment
John Matthews <jjm@aems.demon.co.uk>

"Yatesmeters", Pumpensysteme, intelligente Sensorkarten am Master-Prozessor und ... Forth.

10 Behind the News
Chris Jakeman <cjakeman@bigfoot.com>

Ein erweitertes Editorial über erstaunlich starke Forth-Aktivitäten in der Welt, die "nicht immer gleich offenkundig werden". Zwei neue Forth-Bücher (über Anwendungstechniken und über Lego-Roboter). 166 Forthwrite-Downloads in zwei Monaten. Tom Zimmers Win32Forth wird in der Stärke von

100 MB pro Tag downgeloadet. Roboter in USA und Deutschland. Spezielle Erwähnung der Bemühungen von Ralph Hempel, Martin Bitter und Fred Behringer in Richtung auf "youngsters", Roboter und Forth. Hinweis auf diesbezügliche Veröffentlichungen in der Vierten Dimension.

12 F11-UK FIG Hardware Project
jeremy.fowell@btinternet.com

Jeremy berichtet über Mitteilungen von zwei Mitgliedern, David Abrahams und Jeff Penn, über den Zusammenbau der F11-Karte und ihre ersten Erfahrungen. Ein kleiner fehlender Hinweis im Anleitungsheft war zu ergänzen.

15 F11-UK
FIG-UK-Hardware-Projekt. 47 engl. Pfund, Porto und Verpackung 4 Pfund (auf der Insel nur 2 Pfund), Pygmy-HC11-Forth-Registrierung 25 USDollar. Pygmy-HC11-Forth-Compiler mit allen Quelltexten. Über 100 Seiten Dokumentation. HC11, 8 MHz. 103x100 mm. Mehr bei:
jeremy.fowell@btinternet.com .

16 XML and Forth
Les Kendall <Les@Cyberforth.com>

Der Autor programmiert als freier Mitarbeiter in Visual BASIC, C++ und Forth. Er preist hier die Vorzüge von XML. Daten nicht einfach ad hoc versenden, sondern mit XML-Tags versehen und aufbereiten. In höchstem Maße transferierbar, beispielsweise von Forth in eine Excel-Tabelle und umgekehrt. XML (extensible Markup Language) ist nahe verwandt mit HTML und wird vorwiegend in Web-Anwendungen verwendet. Wird von Microsoft, Oracle, IBM, Sun und vielen anderen Softwarehäusern unterstützt. HTML hat mit der Darstellung von Daten zu tun, wohingegen XML sich um die Daten selbst kümmert. Microsoft hat sein Windows.NET noch etwas zurückgestellt, um mit seinen Anwendungen ganz bestimmt auch XML anbieten zu können.

23 euroForth 2000 - The Report
Howerd Oakford <howerd@inventio.co.uk>

Ein Teilnehmer gibt kurze, aber sehr informative Beschreibungen der Inhalte der einzelnen Vorträge. 23 Teilnehmer, 14 Vorträge, 4 Workshops. England, USA, Österreich, Spanien, Rußland. Deutschland? Meinung des Rezensenten: Wirklich interessant, was man da liest, was auf diesem Fachkongreß geboten wurde und wird. Aber als Anfänger - und auch solche wollen wir ja bei uns immer wieder verstärkt ansprechen - und reiner Amateur wäre man dort wohl kaum am richtigen Platz. Es zeigt sich ein weiteres Mal, daß auch die Vereine in Sachen Forth ihre Berechtigung haben. Nachgelesen werden können die Vorträge unter
http://dec.bournemouth.ac.uk/forth/euro/ef00/

27 Vierte Dimension 4/00
Alan J M Wenham <101745.3615@compuserve.com>

Eine gute Zusammenfassung des Inhalts der Vierten Dimension 4/00. Besonders breiten Raum nehmen die Zeilen über Martin Bitter und die Media-Night in Vaterstetten ein. Ausführlich wird auch über Herbert Finks Lösung des Umlautproblems gesprochen. Alan gelingen zusätzliche Erklärungen mit Anwendung auf die englische Sprache. Ausgezeichnet die von Chris Jakeman angefügte Ankündigung unserer Forth-Tagung 2001.

30 Forth Application Techniques
Robert.Ives@schroders.com

Besprechung eines neuen Forth-Buches von Forth Inc. unter der Autorschaft von Elizabeth D. Rather. 144 Seiten, spiralgebunden, Einführungspreis nur 20 US-Dollar. Basiert auf einem gleichnamigen Kurs, der über Jahre hinweg ständig überarbeitet wurde. Die kostenlose Probeversion von SwiftForth wäre gut, wenn man die Beispiele wirklich zum Laufen bringen möchte. Eine CD wäre empfehlenswert. Wertvolle Ergänzung, aber nichts für den absoluten Anfänger.

32 Letters

4 Briefe an den langjährigen und verdienten Redakteur der Forthwrite, Chris Jakeman. Ed Hersom: Er ist inzwischen hochbetagt, "lehnt sich zurück" und freut sich, daß sein Permutations-Algorithmus wieder zu Ehren kommt (Forthwrite 109). Fred Behringer: Wil Badens EXCHANGE aus Forthwrite 109 geht auch kürzer. Geht es noch kürzer? Garry Lancaster: Über Dave Pochins "blitmode" aus Forthwrite 109. Garry bekam seinen Job in der Firma aufgrund seiner vorweisbaren Erfolge als Forth-Amateur. John Jones: Forth als Maschine, nicht als Sprache. Das ist es, was uns begeistert. Alle anderen Sprachen sind nichts anderes als wirklich nur Sprachen und schränken unser Denken unnötig ein.

CAMs und Forth

Ulrich Paul

Leitershofen

Hi Martin,
hier ein paar Bemerkungen zu CAMs und wo man sie in Forth gut gebrauchen könnte:

Ein CAM (Content Addressable Memory) ist ein Halbleiterspeicher, der genau anders herum funktioniert wie ein normaler Speicher. Während man bei einem normalen Speicher eine Adresse anlegt und am Ausgang die darunter liegenden Daten bekommt, legt man an ein CAM den gesuchten Inhalt (also die Daten) an und man erhält am Ausgang die Adresse unter der sie gefunden wurden. (Ein extra Pin zeigt an ob die Daten überhaupt gefunden wurden und die Adresse auch wirklich gültig ist. Aber das ist ein Detail auf das wir hier nicht näher eingehen brauchen.) Die Geschwindigkeit liegt in dem selben Bereich wie ein normaler Speicher, also, je nach Ausführung, zwischen einigen Nanosekunden und einigen hundert davon. Man kann diese Chips kaskadieren und parallel schalten und so beliebig breite Daten und auch beliebig großen Speicher erhalten. Sie sind in der Regel in SRAM-Technologie gefertigt und ihr Preis liegt bei ungefähr dem Dreifachen eines SRAMS gleicher Größe. Derzeit finden ca. 90% von diesen Chips Verwendung in Routers und Switches (die Dingers, die auch z.B. das Internet zusammenhalten). Könnte man die gewinnbringend in Forth, bzw. einer Forth-Maschine, einsetzen? Meiner Meinung nach ja, aber nur unter bestimmten Umständen.

Unsinn ist ihr Einsatz in folgenden Fällen:

1.) Es wird nur eine bestimmte Applikation ausgeführt und diese Applikation profitiert nicht durch ihre Aufgabenstellung von einem CAM. Profitieren wird aber eine Applikation, die sehr häufig in Tabellen nachschauen muß und wenn sich diese Tabellen regelmäßig ändern.

2.) Es wird meistens im interaktiven Modus gearbeitet. Ganz gleich, wie groß das Dictionary auch ist, ein CAM bringt hier nicht viel. Das bißchen Geschwindigkeit wird viel zu teuer bezahlt. Ein schnellerer Prozessor ist die bessere Wahl.

3.) Die Zeit zum Kompilieren der Applikation ist tragbar.

Das hört sich verdammt danach an, als ob damit praktisch alle Fälle abgedeckt werden. Nicht ganz. Wie schon unter Punkt 1 angedeutet, gibt es Anwendungen, die sehr wohl von einem CAM profitieren können. Man denke da nicht nur an die "klassischen" Fälle der reinen Tabellen-Abfragen. Forth hat ja einiges eingebaut, was andere so nicht haben (oder nicht ganz so). Stellen wir uns doch einmal vor, Adobe kommt auf die Idee "Postscreen" herauszugeben. Das sei ein Bruder von "Postscript", nur eben gedacht, Movies auf den Bildschirm zu bringen. Aus Gründen der Kompatibilität sei Postscreen eine Erweiterung von Postscript. (Wer es noch nicht weiß: PostScript ist ein "verhunageltes" Forth. Gut, es ist nicht ein wirkliches Forth, aber doch dem sehr ähnlich.)

Ja und? Was hat das mit CAMs zu tun? Eine Menge! Nehmen wir eine Bildwiederholrate von 25 pro Sekunde an. Weiterhin kann man wohl davon ausgehen, daß mit weniger als einer Million Vektoren pro Bild kein Mensch mehr zufrieden ist. Somit müssen 25 Millionen Vektoren pro Sekunde bearbeitet werden. Für die Graphikkarte wohl kein Problem. Aber das Postscreen ist keine Vektorgraphik, sondern ASCII-Text! Wir müssen also die Befehle in dem File in solche umsetzen, die die Graphikkarte versteht. Der Einfachheit halber sagen wir einmal, daß jeder Befehl auch einem Vektor entspricht. Das ist zwar sehr vereinfacht, aber hier reicht das allemal. Schaut man sich ein Postscript-File an, dann sieht man, daß die Namen der Befehle eigentlich alle recht kurz sind. Selten ist einer länger als 8 bis 10 Zeichen. Wir nehmen also an, daß Postscreen es auch nicht anders hält. Und jetzt bauen wir unsere Postscreen-Box! Sie besteht im Wesentlichen aus diesen Teilen:

1.) Ein Medium über das wir zeilenorientiert den Postscreen-Text bekommen. Das sei uns gegeben; wie auch immer das ausschaut (Ethernet oder so).

2.) Eine definierte API zur Graphikkarte. Auch dieses interessiert uns hier nicht besonders, da es mit dem Problem an sich (Postscreen-Interpreter) nicht viel zu tun hat. Sicher, eine bessere Graphikkarte nimmt uns Arbeit ab, aber nicht beim Problem der Interpretation, sondern nur bei dem Code, der für jeden Befehl von uns ausgeführt werden muß.

3.) Die Interpreter-Maschine. Das ist der Teil auf den wir uns hier konzentrieren werden. Wie schaut die aus?

Es ist sicher nicht sinnvoll das gesamte Dictionary in CAMs zu legen. Dazu müßten wir das gesamte CAM-Subsystem so auslegen, daß selbst der Befehl mit der größten Länge noch dort hinein paßt. Das verschwendet haufenweise Platz, da durch die Datenbreite des CAM-Subsystems auch die Granularität festgelegt ist. Wenn wir z.B. es auf eine Datenbreite von 10 Zeichen auslegen (also 80 Bits), dann können wir nur in Einheiten von 80 Bits darauf zugreifen und alle Einträge müssen auch an diesen Grenzen liegen. Mit anderen Worten: Wir können nur immer 80 Bits als eine Einheit behandeln. Kein Eintrag darf über diese Grenze hinaus gehen. Es kann also keiner ein bißchen in den einen 80 Bits sein und der Rest in anderen. Also teilen wir das Dictionary in zwei Teile: Der eine enthält alle Worte mit mehr als 10 Zeichen und der Rest liegt im CAM. Den ersten Teil implementieren wir wie gewohnt im normalen RAM des Prozessors. Anhand der Länge eines Names entscheiden wir, ob wir ihn im CAM suchen (oder dort eintragen - sofern noch Platz ist) oder im normalen RAM. Über die Struktur des Dictionarys im normalen RAM brauchen wir hier keine Worte verlieren. Das ist hinreichend abgehandelt, diskutiert und religiös vertreten worden. Aber über das CAM-Subsystem sollten wir schon ein bißchen reden, vor allem, wie man es einbindet.

In dieser Abhandlung lasse ich das Schreiben des CAMs weg, da dies eine Hardwaresache ist und - vor allem - hier nebensächlich ist, da das Verhältnis von Schreib- zu Lese-Operationen so ungefähr 1:100 beträgt. Das Augenmerk richten wir auf die Einbindung des CAM-Subsystems zum Lesen. Wir können wohl mit einiger Berechtigung davon ausgehen, daß selbst Postscreen nicht mehr als 8000 Befehle kennt oder durch temporäre Definitionen haben wird. (Postscript, und damit Postscreen, kann - wie Forth, ach da schau her - neue Befehle on-the-fly definieren.) Unser CAM braucht daher nur 8k "tief" zu sein. (Die "Breite" haben wir oben schon auf 10 Zeichen, also 80 Bits festgelegt. Aber das spielt hier keine Rolle.) Die Chips haben meistens einen Ausgangsbus, der der Tiefe des Chips (der Anzahl der Speicherstellen) entspricht. Bei 8k sind das 13 Bit. Die Chips sind so miteinander verdrahtet, daß uns diese Information reicht (siehe Kasten). In unserem RAM haben wir uns beim Booten eine Tabelle angelegt. Sie hat die Breite des Adressbusses unseres Prozessors (z.B. 16 Bit) und 8k Einträge. Jeder Eintrag enthält die Startadresse der Routine, die für ein bestimmtes Word (das wir im CAM gefunden haben) zuständig ist. Diese Indirektion müssen wir leider machen, da wir halt "nur" ein CAM verwenden. Hätten wir statt dessen ein "fully associative cache" (mit ausreichender Größe!) zur Verfügung, dann könnten wir uns diese Tabelle und das Nachschauen darin sparen. Aber das können wir uns hier einfach nicht leisten. Da kaufen wir uns lieber einen Palm-Rechner, buchen ein Wochenende in London (oder Paris oder ...) und werfen den Palm dort weg - das ist billiger. (Na ja gut, ich lasse ja mit mir handeln: Den Palm verscherbeln wir.)

Nun haben wir die Startadresse unseres Codes, der für einen bestimmten Befehl zuständig ist. Brachte uns das CAM etwas? Ja, denn bei einer Suche im RAM hätten wir wesentlich mehr Zeit verbraucht. Bei einer bloß halbwegs geschickten Implementation des CAM-Subsystems sind wir mindestens um den Faktor 8 schneller. Und das heißt in unserem Fall, daß wir tatsächlich 25 Bilder pro Sekunde darstellen können oder halt nur 3 1/8.

Müssen wir auf "Postscreen" warten bevor sich ein CAM rentiert? Nicht unbedingt. Was ich als Beispiel genommen habe, ist nicht so exotisch. Bei entsprechender Implementation profitieren eine Menge Konverter auch davon. Vor allem solche, die zwischen zwei syntaktisch äquivalenten Räumen vermitteln/übersetzen. Das kann z.B. Deutsch und "Hundhammer-Deutsch" sein (benannt nach einem Bildungsminister kurz nach den 2. Weltkrieg in Bayern. Das klingt etwa so: "I am heavy on wire" oder "you are standing well on the hose" oder "he made himself me nothing you nothing out of the dust". Wen das an eine Birne erinnert hat KEINEN Tick!) Sinnvoller sind andere Anwendungen, vor allem in der Steuerungstechnik. Hier gibt es eine Menge Protokolle, die sich im wesentlichen nur in den Befehlsnamen unterscheiden.

Die "schlauen" Kerlchen von CAMs (nicht die Politiker!) können noch in viel mehr Gebieten eingesetzt werden. Eigentlich setzt nur die Phantasie eine Grenze (ok, derzeit auch noch der Preis). Wem fällt die heißeste Verwendung ein? Ich bin gespannt.

Anmerkung: Leider verwenden die verschiedenen Hersteller die Begriffe "kaskadieren" und "parallel schalten" nicht einheitlich. In dem Zusammenhang hier heißt kaskadieren: Mehrere Chips so zusammenschalten, daß mehr Worte einer gegebenen Breite zur Verfügung stehen. Wenn man also 2 Chips, die je 16 Bit breite Worte haben und davon 4k speichern können, kaskadiert, dann ist das äquivalent zu einem Chip mit 16 Bit Breite und 8k Tiefe. Im Gegensatz dazu steht die Parallelschaltung. Sie bewirkt, daß man mit den o.g. Chips einen äquivalenten Chip von 32 Bit Breite und 4k Tiefe erhält. Kurz und bündig: Kaskadieren erhöht die Tiefe, die Parallelschaltung die Breite.

Sinnvollerweise baut man ein CAM-Subsystem, das breiter ist als ein einzelner Chip (d.h. daß die zu suchende Information breiter ist als das, was ein einzelner Chip anbietet) so auf, daß die Chips parallel geschaltet sind. Was heißt hier parallel? Es heißt, daß wir die Ausgänge aller Chips, bis auf einen, unangeschlossen lassen. Nur die "Match"-Leitungen, die meistens wunderbar zu einem Wired-And zusammenschaltbar sind, werden von allen Chips gebraucht. Auf der Eingangsseite kriegt natürlich jeder Chip die Information, die seinem Anteil entspricht. In unseren Beispiel mit CAMs, die 8k Worte zu je 16 Bit speichern können, kriegt der erste Chip die ersten 2 Zeichen des Befehls, der zweite Chip die nächsten zwei und so fort, ab. Beim Schreiben des CAMs müssen wir (durch die Hardware) sicher stellen, daß alle Teile unserer 80-Bit-"Worte" an den identischen Adressen in den jeweiligen Chips landen. Das klingt hier komplizierter als es ist. Technisch ist es sehr einfach: Beim Schreiben eines CAMs werden die Ausgänge des normalen Lese-Betriebs Eingänge. Wir legen also - wie auch beim Lesen - an den Daten-Eingängen die Information an. Nur legen wir - und nicht der Chip - die Adresse am "Ausgang" an, unter der wir diese Daten speichern wollen. Da wir diese Adresse bei allen Chips gleich angeben (für ein Informations-Wort) wissen wir, daß alle Chips bei einem Match auch die gleiche Adresse als Ergebnis liefern werden und wir daher nur die Antwort eines Chips anschauen müssen.

From the big Teich

Henry Vinerts

Dieser Beitrag unseres langjährigen 'Auslandskorrespondenten' Henry Vinerts aus dem Silicon Valley wurde von Thomas Beierlein ebenso wie Freds Briefe in der letzten Ausgabe der ViertenDimension übersetzt

Lieber Martin,

seit Du für Friederich eingesprungen bist, geht diese Post zuerst zu Dir, mit Kopien an Fritz und Fred. Es ist mein monatlicher Bericht über die Fortschritte in der Silicon Valley Forth Gruppe.

Gestern war das SVFIG-Januar-Treffen. Ich konnte nur an der Morgensitzung teilnehmen, damit wird dies ein kurzer Bericht. Da Dr. Ting mit seinem Vortrag für den Nachmittag eingeteilt war, aber dienstlich unterwegs war, könnte der zweite Teil des Tages wie eine unerwartete Freistunde in der Schule gewesen sein, mit einer Menge Zeit für Geplauder, Geselligkeit, und Spaß und Spiele wie es sich für eine ältere Gruppe von Forth Anhängern geziemt.

Weißt Du, ich habe irgendwo gelesen, daß über 400 unterschiedliche Sprachen rund um die Welt gesprochen werden, aber nur um die 220 haben mehr als eine Million Sprecher. Es gibt ungefähr 200 nordamarikanische Indianersprachen, aber nur 50 haben mehr als 1000 Sprecher. Forth ist wie eine dieser Sprachen. Sie wird überleben, nicht aufgrund von Geld oder Prestige, sondern durch die Kultur die sie repräsentiert, solange es junge Menschen gibt, die ihr folgen wollen. Ich war erfreut zu sehen, daß in der Gruppe der etwa 20 Teilnehmer zwei ziemlich frische Newcomer waren - ich denke beide in den Dreißigern. Das Durchschnittsalter des Restes ... ich will lieber nicht darüber spekulieren.

Ebenso wie Ting wurde seine Kaffekanne vermißt, aber George Perry hatte die "bagels" mitgebracht (--- der Übersetzer und sein Wörterbuch haben keine Ahnung was das ist, es muß also etwas ganz harmloses sein.... ---*s.u.) und wir verbrachten die erste halbe Stunde ziemlich fröhlich, während wir auf John Rible warteten, der sich mit seinem angekündigten Vortrag über 'linear rückgekoppelte Schieberegister' LFSR verspätete. Er kam zu spät, da dort wo er lebt die Kopiermaschinen so veraltet sind, daß er eine Stunde mehr brauchte um die Kopien für uns anzufertigen. Für mich, und vielleicht einige andere, lag das Thema so nah am Herzen wie, z.B. die 14 Arten der konjugierten spanischen Verben. Aber ich muß sagen, John versteht sein Handwerk. Für jeden der etwas über LFSR's wissen möchte empfiehlt John den Anhang F in Clive Maxfield's Buch "Bebop to the Boolean Boogie." (--- Kennt jemand den deutschen Titel? d. Übersetzer :-) ---) Ich habe das Buch, auf John's Empfehlung, vor einigen Jahren gekauft und ich sehe, daß es wohl Leuten wie mir Elektronik beibringen kann, aber bis jetzt habe ich nur den Abschnitt über das vigezimal System (Basis 20) gelesen, welches von den Azteken, Maya's und Kelten benutzt wurde. Zu viele Bücher, zu wenig Zeit!

So, Du siehst, ich habe dieses mal nicht allzuviel von Wert zu berichten. Ich hoffe, daß mit der Forth Gesellschaft e.V. alles in Ordnung ist und ich sollte nach dem 24. Februar wieder etwas haben, worüber ich schreiben werde. Gebt acht!

Henry

(Bagels: ein Gebäck der jiddischen Küche. Ein Kringel aus Hefeteig, der kurz gekocht und dann im Ofen gebacken wird. Die Oberfläche bekommt dadurch eine eigene Konsistenz. Zuckerguss nach Belieben. M.B. :-)

(ANS-Forth)Quellcode oder (Java)Komponente – was denn nun?

Bibliothekskonzepte Teil III

Jörg .Staben

Langenfeld

Jetzt wurde schon soviel über Bibliothekskonzepte gesagt; können wir das nicht mal life und in Farbe sehen?

Das bunte Leben kreist häufig um´s Geld; die Geldanlage kreist um die Rendite und da interessiert häufig die Anlagedauer einer Geldanlage.

Eine Geldanlage soll von heute bis zu einem Enddatum, sagen wir dem 24.09.2001, angelegt werden. Das aktuelle Datum kennt der verwendete Rechner, das Enddatum möchte bitte der Anwender eingeben. So in der Form etwa:

> Heute ist der 14.01.2001 .
>
> Wann wolllen Sie das Geld wiederhaben (Tag, Monat, Jahr)?
>
> Anlagedauer ist 253 Tage.

Wie ich in DE.COMP.LANG.FORTH bereits gesagt habe, nehme ich da gerne eines der gängigen Office-Pakete, hier das Star-Office:

Mit so´nem Office ist sowas ja schnell gemacht... Aber darum geht´s ja hier nicht, es geht um Bibliothekskonzepte. Und nun zuerst FORTH:

Forth - seine Infrastruktur

Wenn man überlegt, was man so braucht, kommt man auf drei Dinge (kennt eigentlich noch einer die Tabakwerbung: Drei Dinge braucht der Mann...):

Wie kommt man an das Tagesdatum ?

Wie läßt man den Anwender das Enddatum der Geldanlage eingeben ?

Wie errechnet man die Differenz in Tagen zwischen Tagesdatum und dem eingegebenen Enddatum?

Das Win32Forth spuckt sofort TIME&DATE aus, das ist schon mal ganz brauchbar.

Weniger brauchbar ist die Suche nach NUMBER INPUT; das Win32Forth schweigt sich erstmal aus, läßt aber den hoffnungsfrohen englischsprachigen Anwender zumindest sofort im Internet nach Tips suchen.

Wenn sich jemand vor Augen führen möchte, was ich immer mit Infrastruktur einer Programmiersprache meine, der probiere bitte mit z.B. GOOGLE als Suchmaschine folgendes aus „Zeicheneingabe in ...“ Und dann für die Pünktchen ... nacheinander FORTH, JAVA und C++ einsetzen.

Dennoch: Im englischsprachigen Netz ein FORTH-Glückstreffer!

Unter http://www.quartus.net/products/forth/ findet sich nicht nur ein FORTH für den PALM Organizer, sondern auch eine Bibliothek von **ANS-Forth-Wörtern** für Zeitumrechnungen auf Basis des gregorianischen Kalenders! Toll!!

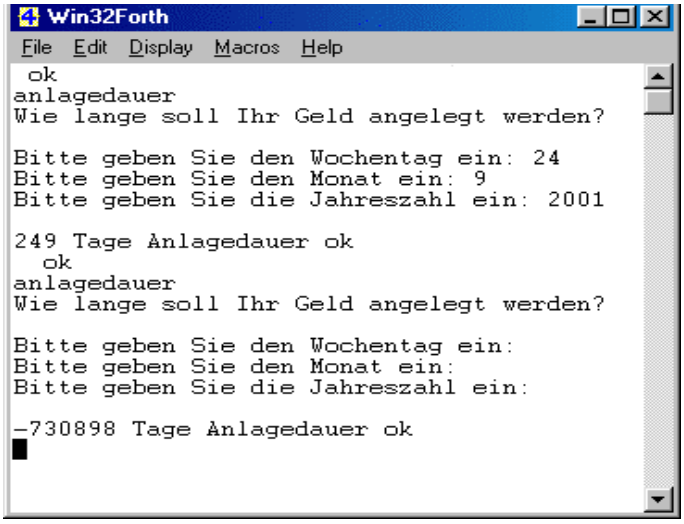
Drei, vier Dateien runtergeladen, mit dem Win32Forth compiliert – geht!

Jungs und Mädels der FORTH-Community, seid dankbar, daß ihr ANS-FORTH und das Internet habt!

Wiederverwenden von Programmcode, Standardisierung und Verfügbarkeit stehen wohl in engem Zusammenhang.

Und nun die Realisierung des Programms und zum allwiederkehrenden Drama der Zahleneingabe in FORTH:

Man muß wissen, daß man eine Zeichenkette in einen Puffer einliest; man muß daran denken, den Puffer jedesmal vorher zu löschen; ebenso liegt die komplette Fehlerbehandlung wie der 31.Februar, das Jahr 20010 oder der 0.Januar beim Programmierer.



Ein Blick auf die Konkurrenz - Java

Mit der Suchmaschine JFIND habe ich recht schnell zwei CalendarBeans gefunden:

Das erste - ITILsCalendarDemo - integriert wunderschön in das VisualCafe 4.1, bringt aber das Forte4Java 1.0 zum Hängen ... weniger schön. Dafür hat das Bean aber ein Funktion diffDate, die verdächtig nach dem aussieht, was wir suchen.

Das zweite CalendarBean (FreeWare) von Kai Toedter integriert wunderbar in Forte4Java 1.0 und ist so´ne Art Zuckerguß für die Java-Klasse ´Gregorianischer Kalender´.

Und damit ist man dann schnell an der Fertigstellung für das gesamte Programm; ich fürchte, deutlich schneller als das Ausprogrammieren der Zahleneingabe in Forth.

Ganz sicher sind auch Komponenten nicht allseligmachend, aber sie helfen weiter.

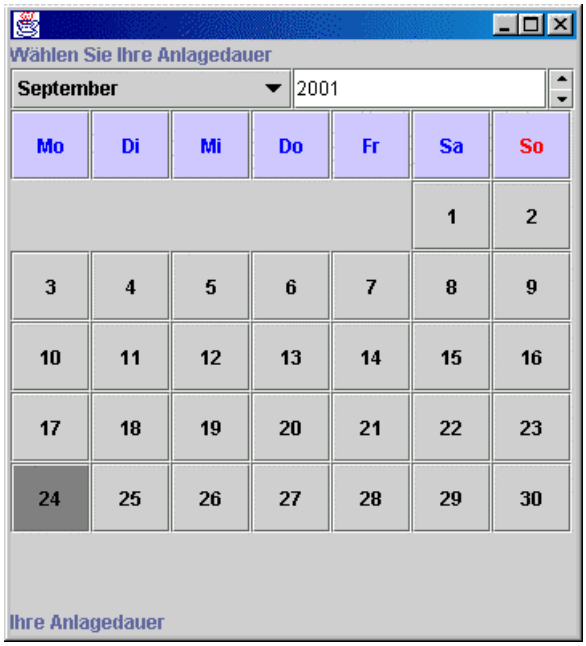
Die Vorteile einer Komponente ergänzen sich auf der Anwender (end-user) und auf der Entwickler-Seite:

Für den Anwender:

Intuitive Anwendung
Bediensichere Anwendung

Für den Programmierer.

Die komplexe Plausibilitätsprüfung der Datumeingabe entfällt



Das Layout der Schnittstelle zum Endanwender (in FORTH mit GOTOXY und CR) entfällt.
Ähem, hüstel: So´n GUI-Outfit für das eigene Programm sieht schon schick aus.

Zum Abschluß ein Ratespiel:

In diesen Text sind drei Abbildungen eingestreut: Welches ist die Bedienoberfläche des Forth-Programms?

Gehaltvolles

Teil 2

zusammengestellt und übertragen von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande
Nr. 24, Februar 2001

De belofte voor 2001
Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>

In allen Vijgeblaadje-Heften von 2001 werden Artikel über das Roboter-Vorhaben zu lesen sein. Das Schema steht schon fest. (Der Rezensent hat das Gefühl, den Grundriß seines Lieblingsroboters, Hank, aus dem Lego-Baukasten vor sich zu haben.) Als Teile werden vorwiegend solche genommen, die man bei Conrad bekommt. Bestimmte Vorgaben werden gemacht: Kommunikation untereinander, kämpfen gegeneinander, Fahrt durchs Labyrinth, selbständig lesen und schreiben, Infrarot-Schnittstelle ... und natürlich ByteForth. Ein Großteil der eigentlichen Ausführung bleibt den Nachbauern und Wettstreit-Teilnehmern überlassen.

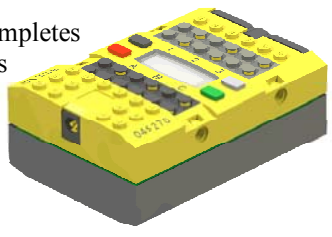
Nieuwe ByteForth releases
Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>

Zwei neue ByteForth-Ausgaben stehen vor ihrer Vollendung: 8051-ByteForth 2.00 und das neue AVR-ByteForth 1.00. CREATE, DOES> und CASE werden aufgenommen, und Worte zur Verarbeitung und Darstellung von 32-Bit-Zahlen.

Das Forth-Treffen fand am 10. Februar 2001 statt. Tagungsort ist neuerdings: Katholieke Vrouwengilde Nederland, Bisonspoor 1204, Maarssenbroek.

LEGO RCX-Verleih

Seit unserm Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX Komponenten, dass ich meine privat eingebrachten Dinge nun anderen, vorzugsweise Mitgliedern des Forth e.V. zur Verfügung stellen kann!
Angeboten wird: Ein kompletes LEGO-RCX-Set, so wie es für 450 DM im Handel zu erwerben ist. Anfragen über die Redaktion an mich.



Martin Bitter

Karatsuba Teil 1

Von Martin Bitter

Mehrhoog

Dieser Artikel stellt in gewissem Sinne eine Fortsetzung aus der Vierten Dimension 1/2001 dar. Dort ging es um eine schnelle Hardwaremultiplikation mithilfe der 'russischen Multiplikation'. Dabei ging es (mir und Fritz Prinz) um die prinzipielle Darstellung des Algorithmus' – nicht unbedingt um die tatsächliche Umsetzung in Hardware, die durch die Verwendung von Tabellen, Einsprungstellen usw. Speicherplatz spart und Geschwindigkeit gewinnt.

Auf der Jahresversammlung 2000 in Hamburg berichtete Ulrich Hofmann über eine Hardware, die u.a. Segelflugzeiten aufzeichnet. Das brauchen Segelflieger als Nachweis. Die zugehörige Software war in Forth geschrieben und verwendet 'lange' Zahlen, deren Wertebereich deutlich über den üblichen 64/32/16 Bit liegt. Meine Frage, ob er die Schulmethode verwende oder sich an Kara... (mir fiel doch der genaue Name nicht mehr ein!) orientierte, brachte zu Tage, dass von den Anwesenden niemand die Karatsuba-Methode (ich habe inzwischen nachgeguckt ;-) kannte, oder sich nicht dazu äußerte.

Roaring Sixties

1963 veröffentlichte der russische Mathematiker A. Karatsuba zusammen mit seinem Kollegen Y Ofman (1) eine Methode 'lange' polynomale Ausdrücke zu multiplizieren. (Mathematiker: Augen zu und durch!) Nun kann man mit etwas Willen unsere 'übliche' Art der dezimalen Zahlnotation im Stellenwertsystem als eine Sonderform der Polynomdarstellung betrachten, wobei als Koeffizienten halt 'nur' 0 – 9 erlaubt sind und sich der ganze Ausdruck auf die Stelle 10 bezieht.

Genug der Theorie, wer's genauer wissen will kann ja bei http://triton.mathematik.tu-muenchen.de/~kaplan/ca/spock/praktika/UrSpock unter dem Stichwort 'Algorithmen für schnelle Polynommultiplikation' mehr nachlesen (solange der Link noch Gültigkeit besitzt ... ja, was du schwarz auf weiß besitzt ...

Schulmultiplikation

```
: D* ( d1 d2 - d3 )      \ doppelt genaue Multiplikation
     >R                  \ high_d1 zum Returnstack
     SWAP R>             \ high_d2 zum Returnstack
     2DUP                \ beide high-Werte kopieren
     UM*                 \ mit evtl. Übertrag multiplizieren
     ROT                 \ low_d2 zum TOS rotieren
     R>                  \ high_d2 vom Returnstack holen
     *                   \ 'normal' multiplizieren (****)
     +                   \ positionsrichtig addieren
     ROT R>              \ low_d1 zum TOS rotieren, hole high_2
     *                   \ 'normal' multiplizieren (****)
     + ;                 \ positionsrichtig addieren; fertig
 R. Zechs Definition von D* (neu kommentiert von M.B)
```

Bei Zech (2) ist nachzulesen, wie Forth-Systeme doppelt breite Multiplikationen durchführen können, wenn der verwendete Prozessor keinen entsprechenden 'mul' Befehl zur Verfügung stellt. Daraus kann man ohne weiteres das Vorgehen für noch 'breitere' Multiplikationen ablesen.

Der dort gezeigte Befehl D* multipliziert zwei doppelt genaue (breite) Zahlen, ignoriert aber die evtl. entstehenden zusätzlichen Ziffern (Zellen, Bits) (in der Übersicht stark umrandet).

		high	low	d1
		high	low	d2
		mid-low	low	low_d1*low_d2
	mid-high	mid-low		high_d1*low_d2
	mid-high	mid-low		high_d1*high_d2
high	mid-high			high_d1*high_d2
high	mid-high	mid-low	low	

Dies geschieht dadurch,dass eine 'einfache' Multiplikation statt einer weiteren doppelt genauen Multiplikation durchgeführt wird. Diese Stellen sind durch (***) gekennzeichnet. Natürlich spielt hier auch der Umgang mit 'signed numbers' eine Rolle, aber für unsere Zwecke reicht diese Betrachtung.

Das Ganze noch einmal mit eingesetzen Zahlen und so, wie wir es in der Schule gelernt haben (alle Kopfrechnungen explizit notiert):

6	5	*	7	3	d1	d2
			1	5	low_d1*low_d2	
	1		8		high_d1*low_d2	
	3		5		high_d1*high_d2	
4	2				high_d1*high_d2	
4	7		4	5		

Zählt man die Elementarmultiplikationen, das wäre im menschlichen Fall das auswendig gelernte kleine 1x1, im Maschinenfall der breiteste MUL-Befehl des Native-Codes (Forth um*), so benötigt eine solche Multiplikation mit zwei zwei-zelligen Faktoren vier Elementarmultiplikationen bei einem vierzelligen Ergebnis.

Es läßt sich leicht erkennen, dass eine weiter Verdoppelung der Rechenbreite, die Anzahl der Elementarmultiplikationen quadriert usw. Eine Multiplikation mit zwei vierstelligen Faktoren benötigt sechzehn Elementarmultiplikationen, eine mit achtstelligen Faktoren schon 64!

Bei den heutigen Prozessortakten mag dies zu verkraften sein. Ist man jedoch in der Verlegenheit, mit **wirklich** langen (großen)

Zahlen rechnen zu müssen, - hier geht es um Zahlen mit mehreren Tausend bis Millionen Ziffern - so erlangt diese quadratische Zunahme des Rechenaufwandes und der damit verbunden Zeit eine große Bedeutung.

BIG.F

Im Lieferumfang von SwiftForth ist ein Paket Big.f (3), das die grundlegenden Rechenbefehle zum Umgang mit großen Zahlen bietet. Die theoretische Obergrenze (soweit die physikalischen Möglichkeiten des Rechners es zulassen) liegt bei 68.719.476.704 Bit (größte unsigned einfachgenaue Number mal Zellengröße in Bit = $7FFFFFF * $100000000). So knapp' 70 Milliarden Bit, das ist schon 'ne Menge Holz!

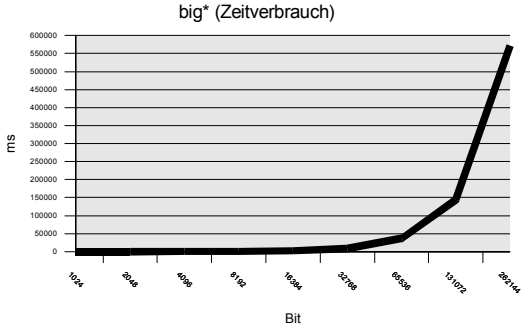
Ich habe auf meinem System (AMD K6, 400 MHZ) gemessen, wie lange das dort implementierte big* für Multiplikationen mit zunehmend größer werdenden Zahlen benötigt:

Zellen	Bit	Zeit in ms
32	1024	0
64	2048	50
128	4096	110
512	8192	550
1024	16384	2250
2048	32768	8960
4096	65536	36250
8192	131072	143130
16384	262144	572330

Selbst bei den immanenten Ungenauigkeiten der Zeitmessungen unter Windows/PC ist die Tendenz doch eindeutig und bestätigt die theoretischen Überlegungen.

Noch augenfälliger sichtbar wird der Zusammenhang bei einer grafischen Darstellung:

Größere Faktoren als 16484 Zellen (262144 bit) wollte ich nicht ausprobieren, immerhin - dieser Artikel soll noch in die zweite Ausgabe der Vierten Dimension 2001



Karatsuba

Mit dem Karatsuba-Algorithmus werden 'breite' Zahlen so multipliziert (vgl. rechte Spalte oben):

Mehreres ist von Bedeutung. Die Zwischenprodukte der beiden High- bzw. Low-Komponenten werden zum einen in die Quadranten 4 und 3 sowie in die Quadranten (Spalten) 2 und 1 geschrieben. Die gleichen Zwischenprodukte werden je

		high	low	d1
		high	low	d2
		mid-low	low	low_d1*low_d2
high	mid-high			high_d1*high_d2
	mid-high	mid-low		low_d1*low_d2 (Kopie!)
	mid-high	mid-low		high_d1*high_d2 (Kopie!)
	mid-high	mid-low		(h_d2-l_d2)(l_d1-h_d1)
high	mid-high	mid-low	low	

ein zweites Mal verwendet und einmal in die höherwertige Richtung (hier nach links) geshiftet, das ist das Produkt der beiden low-Teile – ein anderes Mal in die niederwertige Richtung (hier nach rechts) geshiftet, das ist das Produkt der beiden high-Teile.

Nun werden beide Faktoren in der Mitte getrennt und die entstehenden Zahlen voneinander subtrahiert, wobei bei einer Subtraktion noch Minuend und Subtrahend vertauscht werden. (6 – 5 = 1) bzw. (3 – 7 = -4) Die Differenzen werden multipliziert und das Ergebnis positionsrichtig (Quadrant 2 und 3) addiert.

Fertig!

Auch hier wieder zur besseren Übersicht ein Zahlenbeispiel:

6	5	*	7	3	d1	d2
			1	5	low_d1*low_d2	
4	2				high_d1*high_d2	
	1		5		low_d1*low_d2 (Kopie	
	4		2		high_d1*high_d2 (Kopie!)	
	1			-4	h_d1-l_d1	h_d2-l_d2
			-4		(h_d1-l_d1)	(l_d2-h_d2)
4	7		4	5		

(Die weiß unterlegten Felder enthalten Zwischenergebnisse, die nicht mit addiert werden!)

Ganz langsam und ruhig öfter nachrechnen! Es stimmt!

Nachzählen der Elementaroperationen bei zweistelligen Operanden ergibt,

für die Schulrechnung: 4 mul und 4 add
für Karatsuba: 3 mul und 3 add 2 sub

Und das soll von Vorteil sein?

Nun ja. Wie man's sieht. Während die Steigerung des Rechenaufwandes bei der Schulmultiplikation mit dem Quadrat einhergeht, also der Exponent eine Zwei ist, steigt der Rechenaufwand beim Karatsuba-Algorithmus 'nur' mit einem gebrochenen Exponenten von ca. 1,6!

Und das wirkt sich ab einer bestimmten Größe aus!

Ich habe den Karatsuba-Algorithmus in big.f eingebunden und erreiche damit folgende Zeiten: (s.h. Tabelle).

Bits	kara*	big*
1024	0	0
2048	0	50
4096	50	110
8192	220	550
16384	770	2250
32768	2310	8960
65536	6920	36250
131072	20980	143130
262144	67136	572330

Der Vergleich macht deutlich, dass das übliche Vorgehen zumindest ab Zahlen über 1024 Bit langsamer ist. 'Kleinere' Werte entziehen sich momentan der Messgenauigkeit. Aber 1024 Bit das ist schon eine Zahl mit ca. 160 Dezimalstellen oder genau 256 Hexadezimalstellen.

Selbermachen?!

Für Selbermacher kann der Quelltext 'nkarad.f' ein Ausgangspunkt sein. Das dort definierte karad* kann zwar nur aus zwei 32 Bit Zahlen das 64bittige Produkt bilden, aber es zeigt vieles, das auch in dem 'ausgewachsenen' Karatsuba-Algorithmus von Bedeutung ist:

Karatsuba ist für natürliche Zahlen gedacht, der Umgang mit vorzeichenbehafteteten Werten bedarf deshalb einiger Vorkehrungen (vgl. u+ , u-). Dies wird besonders im dem ELSE-Teil der Verzweigung deutlich. Sobald man den 'üblichen' Zahlenbereich verlässt wird's kompliziert, jedenfalls geht mir das so.

karad* ist fast reines ANSI, es sollte in jeder entsprechenden Umgebung laufen (4dup = 2over 2over).

Referenzen

Ein wichtiger Tipp! Überprüfen Sie im Falle eines Falles ihre Referenzmultiplikation. Denn wer möchte schon 'händisch' nachrechnen, ob 154,642,778,830,705,552 * 5,152,986,184,376,334,438 wirklich 43,198,523,253,970,83613,008,581,825,954,728,800 als Ergebnis hat?

Karamult.exe aus dem berühmten Pi-Buch (4) ist fehlerbehaftet (Es hat mich ziemlich viel Überwindung gekostet, das zu glauben!) Empfehlen kann ich das Paket big.f, das dem SwiftForth beiliegt.

Mit ihm hat man alle Routinen zur Hand, die man benötigt, um lange Zahlen einzulesen und auszugeben (nur im Hexmodus kann ich ohne so etwas einigermaßen 'sehen' ob Ergebnisse stimmen), lange Zahlen zu addieren und zu subtrahieren und es ist zuverlässig.

In der nächsten Vierten Dimension werde ich zeigen, wie ich den Karamult-Algoritmus in Big.f eingebunden habe.

Warum erst in der nächsten Vierten Dimension?

> Der Redakteur freut sich, dass es mindestens einen Artikel für die Ausgabe 3/2001 gibt. (**Das ist ein Wink mit dem Zaunpfahl!**)
>
> Ich erhoffe mir, dass jeder, der das karad* sieht laut ausruft: „Das kann ich besser!“ und das hier zeigt! (Dürfte nicht all' zu schwer sein ;-)
>
> Vielleicht ergibt sich ja eine 'Diskussion', die in den Fortsetzungsartikel mit einfließen kann?
>
> Ich habe auf diese Art noch Zeit an Code zu schrauben, obwohl ja Eleganz (nach Bolzman) nur etwas für Schneider ist.

Der Karatsuba-Algorithmus hat mich für eine ganze Weile fasziniert. Zum einen, weil er händisch (Fachausdruck der Mathematiker, hab' ich inzwischen gelernt ;-) zu bewältigen ist und weil er mathematikgeschichtlich gesehen trotz seiner Einfachheit (in der Anwendung!) recht jung ist.

Zahlen, Ziffern, Digits (Begriffe für den Hausgebrauch)

Zahlen werden durch Ziffern dargestellt, Abhängig vom jeweiligen System kann die gleiche Zahl durch unterschiedliche Ziffern dargestellt werden. So sind A, 10 und 1010 unterschiedliche Darstellungen der gleichen Zahl. (Sedezimal, dezimal, binär)

Ziffern sind sozusagen die Buchstaben der Zahlen. Unterschiedliche Zahlsysteme haben per Konvention und Historie unterschiedliche Ziffern: Üblich sind 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, teilweise A, B, C, D, E, F. Der Vorrat an Ziffern ist (in der Realität) beschränkt, der Zahlenraum (idealiter) unendlich.

Digits Rechnerfamilien haben einen Vorrat von mathematischen (Integer) Befehlen, der sich auf einige wenige festgelegte Zahlenräume beschränkt. Dies sind 8-bit, 16-bit und 32-bit Zahlenräume (seltener 64-bit). Ein Prozessor kann 'native' mit diesen Zahlen umgehen. Bei zusammengesetzten Rechnungen mit größeren Zahlen 'benutzt' der Programmierer solche einzelnen Zahlen, so wie eine einzelne Ziffer beim händischen Rechnen. In Forth ist ein solches Digit (fast) mit einer Zelle (cell) gleichzusetzen. Big.f verwendet 31 bit breite Digits, also nur die positive Hälfte einer 'normalen Zelle.

Literatur:

1) A. Karatsuba und Y. Ofman, 'Multiplication of multidigit numbers by automata', Soviet Physics-Doklady 7, S. 595-596, 1963

2) Ronald Zech: Forth 83, S. 69-70, 1987

3) ftp://ftp.taygeta.com/pub/Forth/Scientific/big.fth

4) Jörg Arndt und Christoph Haenel: PI, Algorithmen, Computer, Arithmetik. ; Springer 2000²

```
\ noch'n karad*
: u+ ( carry n n -- carry n )
     0 tuck d+                  \ unsigned Plus mit Carry (hig-level adc)
     rot + swap ;

: u- ( n n -- n borrow )        \ unsingned Minus mit borrow-Flag
     2dup u<
     IF 1 swap 0 d- drop 1
     ELSE - 0
     THEN ;

: karad* ( d d  -- q )          \ multipliziert zwei doppelt genaue Zahlen u und v
                                \ Ergebnis ist vier Zellen breit
      4dup                      \ Kopien anlegen
      rot um* >r >r             \ erstes Teilprodukt zum Returnstack (high*high)
      um* >r >r                 \ zweites "                           (low*low)
      swap u-                   \ erste Differenz
      dup 1 =                   \ gab's einen Überlauf?
        IF drop negate -1       \ absoluten Wert herstellen und Vorzeichen merken
        THEN
      swap                      \ Vorzeichen nach 'unten'
      2swap                     \ Faktor zwei holen
      u-                        \ zweite Differenz
      dup 1 =
        IF drop negate -1
        THEN                    \ s.o.
      -rot                      \ Vorzeichen nach unten
      um*                       \ drittes Teilprodukt (mixed)
  \ -------- alle Teilergebnisse fertig; jetzt: stellenrichtig addieren! -------------
      2swap =                   \ Vorzeichen holen und vergleichen
      IF                        \ beide Vorzeichen gleich --> Produkt positiv
                                \ ganz 'normal' addieren
         R@ -rot                \ low von Produkt-ll holen nach unten
         0 rot                  \ low Mix-Produkt holen, 0 Carry vorbesetzen
         r>  u+                 \ low von Produkt-ll holen und addieren
         R> R@ swap >R          \ NOS vom Returnstack holen, low von Produkt-hh
         u+                     \ mit Carry addieren
         r@ u+                  \ hig von Produkt-ll addieren (Quadrant 2 fertig)
         -rot +                 \ nach 'unten' Carry und hig von Mix-produkt addieren
         0 swap                 \ neues Carry vorbesetzen
         r> u+                  \ high von Produkt-ll addieren
         R> u+                  \ low von Produkt-hh addieren
         R@ u+                  \ low von Produkt-hh addieren
         swap                   \ Carry zum TOS
         r> +                   \ high von Produkt-hh addieren
      ELSE                      \ drittes Produkt (mixed) ist negativ
         R> dup                 \ low von Produkt-ll holen; kopieren und shiften
         0 swap                 \ Carry vorbesetzen
         R> R@ swap >R          \ NOS vom Returnstack holen, low von Produkt-hh
         u+                     \ mit Carry addieren
         r@ u+                  \ hig von Produkt-ll addieren (Quadrant 2 fertig)
         0 rot                  \ Spalte wechseln Carry wird Summand
         r> u+                  \ high von Produkt-ll addieren
         R> u+                  \ low von Produkt-hh addieren
         R@ u+                  \ low von Produkt-hh addieren
         swap                   \ Carry zum TOS
         r> +                   \ high von Produkt-hh addieren
     \ ----------------   jetzt kommt die Subtraktion ------------------------------
         >R >R                  \ high (doppel) retten
         2swap                  \ Mix-Produkt hochholen
         -rot                   \ high-Teil vom Mix-Produkt nach unten legen
         u-                     \ low-Mix von high Produkt-ll abzabziehen
         rot                    \ hig von Mixprodukt holen; (Spalte 2 fertig)
         +                      \ borrow addieren ( --> (-) + (-) = -
         r> swap u-             \ vorl. Ergebnis dritte Spalte holen; borrow abziehen
         r> swap u-             \ hig-high dazu addieren (Borrow kann nicht mehr auftreten)
         drop                   \ überflüssiges borrow wegwerfen
      THEN
 ;

Quelltext: nkarad.f
```

Metarätsel - Spielereien mit dem Allbegriff

Fred Behringer
<behringe@mathematik.tu-muenchen.de>

Ein notorischer Rätselsteller einer bekannten Forth-Zeitschrift verspricht im Namen des Direktoriums jedem Mitglied für jeden gelösten Punkt des folgendes Rätsels eine Beitragsbefreiung für ein ganzes Jahr:

Rätsel: Das Verzeichnis V meines Forth-Systems enthält eine beschränkte Anzahl von Colon-Definitionen, solche, die per RECURSE konstruiert wurden und sich folglich selbst enthalten, oder/und solche, die sich nicht selbst enthalten. Man erweitere V um eine Colon-Definition A, die alle solche und nur solche Colon-Definitionen von V enthält, die sich nicht selbst enthalten. (1) Wieviele Colon-Definitionen kann A höchstens enthalten? (2) Wieviele Colon-Definitionen muß A mindestens enthalten? - Ende des Rätsels

Es kamen wütende Proteste. Einer drohte mit Austritt. Was war geschehen? Welche durch Streichung einiger Wörter leicht abgeänderte Rätselformulierung rettet den Rätselsteller vor dem Gesteinigtwerden - und läutet den Ruin der betreffenden Forth-Gesellschaft ein? Man gebe alle im Satz mit "Man erweitere V" wesentlich verschiedenen möglichen Streichungen, also alle Lösungen des Rätselrätsels, des Metarätsels, an.

Für Spätentschlossene:

Es ist nie zu spät, sagt ein Sprichwort. Falls Sie diese Vierte Dimension in der Hand halten und es ist schon Sonntag, der 29. April, dann hat das Sprichwort nicht recht: Die Jahrestagung der Forth Gesellschaft e.V.ist schon (so gut wie) vorbei.
Falls es aber erst Donnerstag der 26. oder Freitag, der 27. ist – nun vielleicht reicht es ja noch etwas von der Tagung mitzubekommen. Ein Anruf bei Martin Bitter lohnt auf jeden Fall.Aus dem vorläufigen Tagungsprogramm:

Vorträge (nicht vollständig):

Heinz **Schnitter**	Programmieren in einem verteilten System
Fred **Behringer**	Lego-Roboter und arithmetisierte Logik in Forth
H. **Eckes**	Forth Briefmarke
Klaus **Zobawa**	interessante Projekte aus derMedizintechnik
Chris **Jakeman**	WebForth ein HTML-basiertes (Forth) Kurs
Wolfgang **Allinger**	Erfahrungen mit dem-QuartusForth
...	

Workshops (Vorschläge):

Lecht(z)zeit! Lego programmieren in Echtzeit! Eine vorgegebene Aufgabe, in beschränkter Zeit mit Partner und einem Lego-RCX Baukasten lösen.
Globalisierung: Zusammenschluss oder verstärkte Zusammenarbeit (in welcher Weise)?
Nachwuchssorgen: Spielerisch lernen (Lego-Roboter) und Forth-Tutorials im Netz (WebForth).
Sensoreigenbau aus Legosteinen: Ein mathematisches Problem.

Aus dem Rahmenprogramm, Auswahlliste:

Rundflug über das westliche Ruhrgebiet (auf eigene Gefahr, runter komm'n se' immer ;-)
Schloss Moyland: Joseph Beuys - Künstler vom Niederrhein
Kriemhild Mühle, mittelalterliche High-Tech, mechanische Regelkreise
APX- Der archäologische Park in Xanten, sehen wie die Römer lebten!
Landschaftspark Duisburg-Nord – eine Stahlhütte zur Besichtigung freigegeben
Otto Pankok Museum – eine besinnliche Wanderung durch die niederrheinische Waldlandschaft

Die Vereinstagung findet am Sonntag Morgen gegen $9^{\underline{00}}$ Uhr statt.

Tagungsbeginn ist Freitag Nachmittag gegen $16^{\underline{00}}$ Uhr, Ende am Sonntag gegen $12^{\underline{00}}$ Uhr. Also: Ist nicht doch noch Gelegenheit vorbeizukommen? (Ein Anfahrtskizze finden Sie unter www.Akademie-Klausenhof.de)

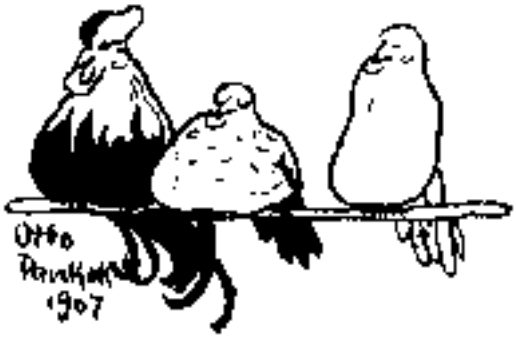
Meine Telefonnummer steht im Impressum.

Bis die Tage, auch!

Impressionen vom Niederrhein

Kriemhildmühle bei Xanten

Schloss Moyland bei Kleve



Otto Pankok: Jugendwerk

Schloss Myoland im Nebel

Römisches Amphitheater mit Lastkran

An die Autoren, Ausblick

An die Autoren:

Bisher haben es alle Editoren/Redakteure geschafft, jeden eingereichten Artikel technisch umzusetzen und in die Vierte Dimension zu platzieren.

Dennoch können Sie uns (mir) die Arbeit ein wenig erleichtern: Verwenden Sie ein gängiges Format. Falls Sie Befürchtungen haben, Ihre Datei sei für uns (mich) kaum zu bearbeiten, schicken Sie diese einfache etwas früher, der Editor wird sich im Zweifelsfall schon bei Ihnen melden.

Senden Sie Ihre Datei(en) in zwei Formaten - eines wird schon passen.

Senden Sie eingebettete Grafiken gesondert, d.h. als einzelne Dateien. Bedenken Sie, dass die Vierte Dimension (leider) 'nur' schwarz weiß erscheint. Konvertieren Sie Ihre Grafiken in Graustufen und machen sie sich zu Hause einen Eindruck, das hilft Enttäuschungen zu vermeiden.

Machen Sie deutlich, wo im Text ihre Grafiken, Abbildungen und Quellfileauszüge erscheinen sollen.

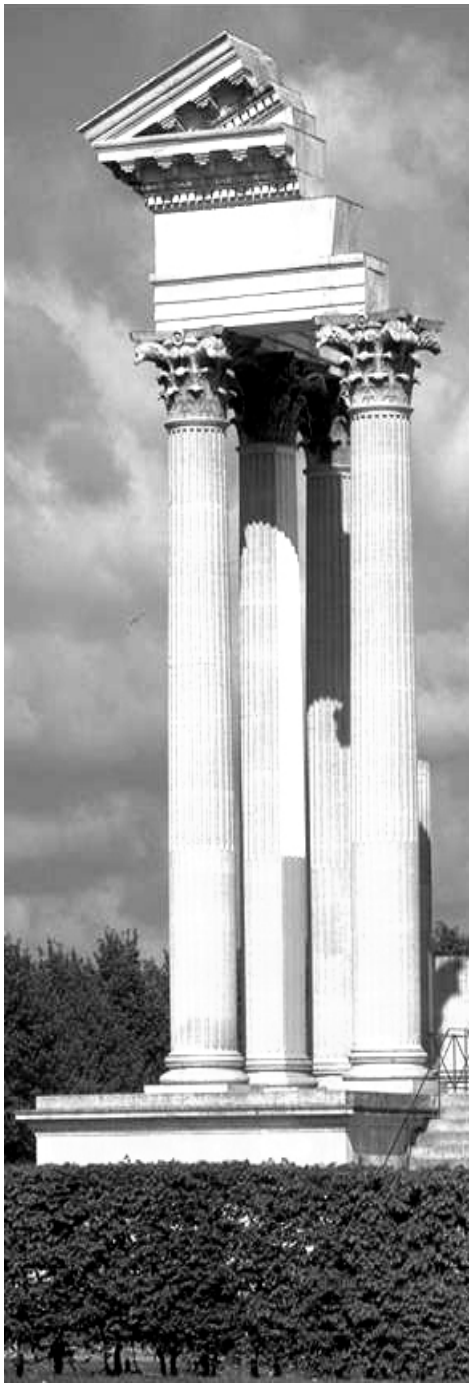
Beachten Sie ebenfalls, dass die Vierte Dimension zweispaltig erscheint. Wenn Sie besondere Wünsche zum Layout haben, senden Sie eine anschaubare Version (WYSIWYG) mit, an der wir (ich) uns orientieren können.

Dank der Rechtschreibreform weiß es jetzt fast jeder: Rechtschreibung ist Konvention ohne Gesetzescharakter! Nur Lehrer und Schüler sind per Gesetz zur Rechtschreibung verpflichtet. In der Vierten Dimension schreibt jeder nach seinem Gusto! Aber auch hier gilt: Zu viel Freiheit schadet nur. Solange die Redaktion so dünn besetzt ist, wie zur Zeit, kann ein Lektorat nicht übernommen werden (was nicht daran hindert, scheinbar offensichtliche Tippfehler zu korrigieren oder mal in dem einen oder anderen Punkt nachzufragen).

Falls Sie Quelltext auszugsweise erörtern, senden Sie - soweit vertretbar - einen vollständigen Quelltext zur Redaktion oder nennen Sie eine anderweitige Bezugsmöglichkeit.

Das Wichtigste: Schreiben Sie - ja auch Sie!

Die jeweils gültige Redaktionsadresse steht im Impressum.



In der nächsten Vierten Dimension lesen Sie voraussichtlich:

Teil 2 des Karatsuba Artikels

Bericht über eine Hardwarebastelei mit dem Lego-RCX

Boot-Mechanismus bei P21/F21 und Boot-Image

Sensoren und arithmetisierte Logik

Ein etwas 'anderes' Festkommapaket

Forth-Gruppen regional

Hamburg **Küstenforth**
Klaus Schleisiek
Tel. 040 / 375008-13 g
kschleisiek@send.de
Treffen jeden 4. Freitag im Monat
16:30 Uhr, Fa. SEND, Stubbenhuk 10
20459 Hamburg

Moers **Friederich Prinz**
Tel.: 02841-58398 (p) (Q)
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: (fast) jeden Samstag,
14:00 Uhr, **MALZ, Donaustraße 1**
47443 Moers

Mannheim **Thomas Prinz**
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-920 185 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München **Jens Wilke**
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im
Monat, **China Restaurant XIANG**
Morungerstraße 8
München-Pasing

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)
micro@forth-ev.de

Gruppengründungen, Kontakte

Fachbezogen 8051 ... (Forth statt Basic, e-Forth)
Thomas Prinz
Tel.: 06271-2830 (p)

Forth-Hilfe für Ratsuchende

Forth allgemein
Jörg Plewe
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02173-75708 (p)

Karl Schroer
Tel.: 02845-2 89 51 (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4	Rafael Deliano Tel./Fax: 089 -841 83 17 (p)
FORTHchips (FRP 1600, RTX, Novix)	Klaus Schleisiek-Kern Tel.: 040 -375 008 03 (g)
F-PC & TCOM, Asyst (Meßtechnik) embedded Controller (H8/5xx//, TDS2020, TDS9092), Fuzzy	Arndt Klingelnberg, Consultants akg@aachen.kbbs.org Tel.: ++32 +87 -63 09 89 pgQ (Fax -63 09 88)
KI, Object Oriented Forth, Sicherheitskritische Systeme	Ulrich Hoffmann Tel.: 04351 -712 217 (p) (Fax: -712 216)
Forth-Vertrieb	volksFORTH / ultraFORTH RTX / FG / Super8 / KK-FORTH Ingenieurbüro Klaus Kohl Tel.: 08233-3 05 24 (p) Fax : 08233-99 71 mailorder@forth-ev.de
Forth-Mailbox (KBBS)	0431-533 98 98 (8 N 1) Sysop Holger Petersen hp@kbbs.org Tel.: 0431-533 98 96 (p) bis 22:00 Fax : 0431-533 98 97 Helsinkistraße 52 24109 Kiel

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen?

Schreiben Sie einfach der VD - oder rufen Sie an - oder schicken Sie uns eine E-Mail !

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeistzeiten
g = geschäftlich

Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

Industrie-Denk-Mal