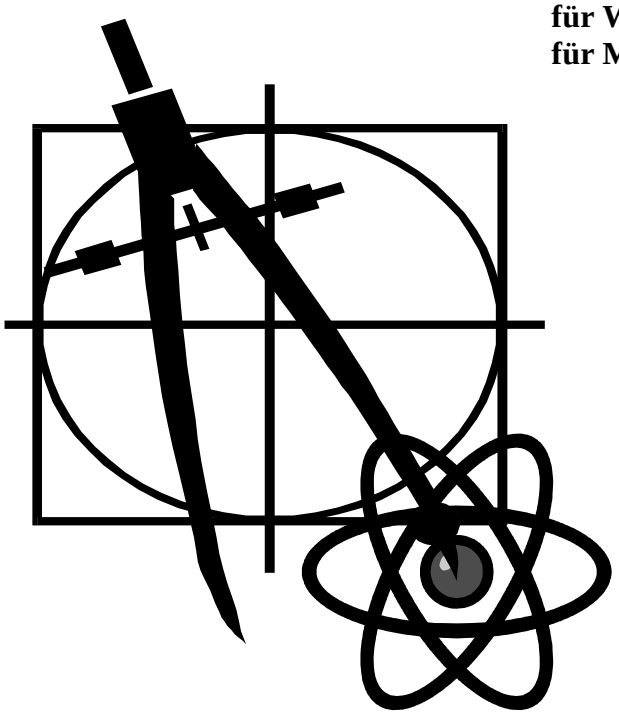
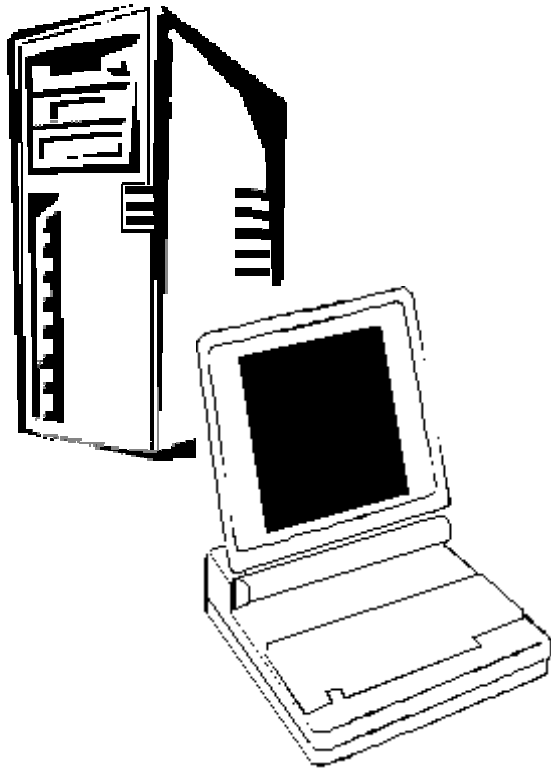
VIERTE DIMENSION

1 / 2004

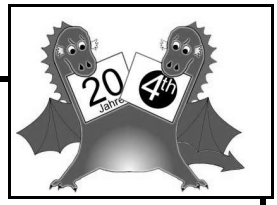
Das FORTH-Magazin

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





In dieser Ausgabe:



20. Jahrgang - **Ausg. 1/2004** - 4,00 €

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D-22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH-Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmeßtechnik und bauen z.Z. eine eigene Produktpalette auf.

Know-How Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential-Dosierwaagen, DMS-Messverstärker, 68000 und 68HC11 Prozessoren, Sigma-Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX-Verleih

Seit unserem 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 der Forthgesellschaft e.V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO-RCX-Set, so wie es für ca. 230,- € im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO Steine.

Anfragen bitte an

Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist ‘narrensicher‘!

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -63 09 88)
Waldring 23, B-4730 Hauset, Belgien
akg@aachen.kbbs.org

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette HighSpeedDuplicating, Tonband, (engl.) Dokumentationen und Bedienungsanleitungen.

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690-380
Fax: 02461/690-387 oder -100
Karl-Heinz-Beckurtz-Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug- und Sondermaschinen, Fuzzy Logic.

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Budapester Straße 80 a D-18057 Rostock
Tel.: (0381) 46 13 99 10 Fax: (0381) 4 58 34 88

PC-basierte Forth-Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software-/Systementwurf. Mehr als 15 Jahre Erfahrung.

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

Tel.: (+Fax) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1-60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX 2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: - 9971
Postfach 1173
D-86404 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Inhalt

Diese Ausgabe der VD wird vermutlich ca. vier bis sechs Wochen nach dem Erscheinen der Druckausgabe im Internet auf der Web-Seite der Forthgesellschaft e.V. veröffentlicht.

http://www.forth-ev.de

Eine PDF-Version dieser Ausgabe wird ab dem Zeitpunkt der Veröffentlichung im Internet ebenfalls zur Verfügung stehen. Bitte wenden Sie sich hierzu über die oben angegebene Adresse an den Webmaster der Forthgesellschaft e.V. oder an die Redaktion der „Vierte Dimension“. *fep*

In der nächsten Ausgabe finden Sie voraussichtlich:

- SCRECs in die Struktur geschaut
- den Code der nicht-deterministischen Maschine
- Lineare Interpolation in Tabellen
- Neues aus der RCX Ecke (Holons Objekte)
- OCR Eingabegerät selbst bauen

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 19 02 25
80602 München
Tel.: (0 89) 1 23 47 84
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

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

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: (0 28 41) 5 83 98
E-Mail: ***VD@FORTH-EV.DE***

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Liebe Leser,

20 Jahre Forthgesellschaft, 20 Jahre „Vierte Dimension“, 20 Jahre mehr Höhen als Tiefen und 20 Jahre Spaß an Forth, Spaß an und mit den Mitgliedern der Forthgesellschaft und Spaß an den vielfältigen Aufgaben, die auch in einem Verein wie dem unseren bewältigt werden wollen – das sind 20 mal 365 Gründe, zu feiern! Und darum wird die diesjährige Tagung der Forthgesellschaft eine ganz besondere Tagung werden. Die Vorbereitungen laufen bereits auf vollen Touren, und ich selbst hoffe sehr, möglichst viele von Ihnen dort persönlich begrüßen zu können.

Können Sie heute noch sagen, was Sie konkret in diesen 20 Jahren an Forthigem bewegt hat, was Sie ausprobiert haben, womit Sie experimentiert haben und was Sie letztlich als „weniger spannend“ betrachtet und zur Seite gelegt haben? Erinnern Sie sich noch an meinen Bericht über das NAXOS? Das war ein Forth-Compiler, in Turbo-Pascal geschrieben, ohne eigenen Interpreter, aber geeignet, ziemlich flotte und gut optimierte DOS-Executes herzustellen. Arbeiten wollte niemand mit NAXOS. Immerhin hätte er aber sehr gut als Beispiel für die Implementierung eines Compilers dienen können, denn seine Quellen waren, meine ich, für jedermann offen zugänglich. Ähnliches gilt für den TCOM, den Target-Compiler von Tom Zimmer. Mit dem TCOM liessen (lassen) sich mit F-PC oder ZF erstellte Forth-Codes zu sehr kompakten DOS-Executes binden. Ich habe lange nichts mehr vom TCOM gehört. Arbeitet noch Jemand damit?
Wolf Wejgaards HOLON ist auch so ein vergessenes Glanzstück. Eine ungeheuer gut organisierte, quasi zur Ordnung in den Gedanken und in der Arbeit zwingende Entwicklungsumgebung mit Forth für verschiedene Targets – hat mich vom Fleck weg fasziniert. Leider ist es mir nie gelungen, diese Faszination auf Mitstreiter zu übertragen. Aber bei allem, was mich selbst am HOLON noch stört, ist dies das Forth, mit dem ich aktuell wieder „bastele“ und meinen PC, den LEGO RCX und mich ausprobiere. Trotz der bunten Windows/Linux Welt arbeite ich immer noch am liebsten in der „guten“ alten DOS-Umgebung, von der ich schwer zu überzeugen war, weil ich mich an CP/M 2.x gewöhnt hatte und erwartete, daß MSDOS mindestens ebenso stabil läuft. Das tat DOS anfangs allerdings gar nicht. Unter CP/M habe ich übrigens mit dem F83 hantiert.

CP/M und F83 – das ist wirklich lange her. 20 Jahre scheinen im Flug vergangen zu sein. Aber schön war es, weil Forth (fast) immer mit dabei war. Und wie war das bei Ihnen?

Ihr

Friederich Prinz

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. *fep*

Die Forthgesellschaft wird durch ihr Direktorium vertreten:

Prof. Dr.	Fred Behringer	Kontakte: Direktorium@Forth-ev.de
Dr.	Ulrich Hoffmann	
Dipl.Inf.	Bernd Paysan	

Betreff: VD Leserbrief / Aufruf des Festkomitees
Von: Michael <michael.kalus@onlinehome.de>

Liebe Mitglieder der Forth-Gesellschaft,

Ich hoffe, ihr hattet ein frohes Fest, und ich wünsche ein gutes neues Jahr.

Wie ihr vielleicht schon wisst, wird dieses Jahr unsere 20. Forth Jahrestagung sein. Sie findet diesmal statt im Hotel Schützenhof in Burgstaaken auf der Insel Fehmarn vom 16. - 18. April 2004. Ein Festessen, Reden und Ehrungen zu diesem Anlass sind schon geplant.

Doch wir - das Festkomitee - sind überzeugt: Forthler können Besonderes. Daher brauchen wir Eure Hilfe. Bitte denkt Euch neben Beiträgen zur Tagung diesmal auch eine Darbietung aus, für den Festabend am Samstag.
Bitte mailt Euren Vorschlag an:

festkomitee <forth04-festkomitee@honkbude.org>

Herzlichen Dank, Euer

M. Kalus

PS: Ich suche dazu auch Bilder von den alten Tagungen, bringt mit, was ihr so habt. Danke. Wir machen eine Pinwand damit und können scannen.

Von: Rafael_Deliano@t-online.de (Rafael Deliano)

a) Paul schreibt in 4/2003, daß er jemand braucht, der ihm Freihandzeichnungen in Grafik umwandelt. Das kann ich machen. Bräuchte sie vorzugsweise als gif-scans per email. Und kann auch previews wieder als gif per email liefern. Alternativ ist zwar Fax auch möglich, aber mühsam.

b) Offensichtlich ist die Wiedergabe der eps-Bilder in 4/2003 nicht üppig gewesen. Adobe Illustrator exportiert natürlich auch andere Formate wie tiff usw. Welche Formate verdaut das verwendete DTP-System vorzugsweise ?

c) Unter "Adressen & Ansprechpartner" ist immer noch "Arbeitsgruppe MARC4" aufgeführt. Die kann man löschen. Man kann allerdings die Leser unter Kurzmeldung darauf hinweisen, daß vom MARC4-Newsletter das Transponder-Sonderheft als scan auf

http://www.embeddedFORTH.de verfügbar ist.

MfG *JRD*

Zu c) - wie auf der vorletzten Seite nachvollzogen werden kann, ist dieser Vorschlag bereits umgesetzt.

Fep

Betreff: VD 4/2003 --> VD 1/2004
Von: Carsten Strotmann <carsten@strotmann.de>

Hallo Fritz,

ich habe heute die neue VD bekommen und schon fast ganz durchgelesen. Ist diesmal wirklich sehr interessant geworden (insb. Traget-Compiler, 4 gewinnt und Fletcher Prüfsumme).

Für die kommende Ausgabe habe ich schon eine Artikel-Idee. Die Mannheimer FORTH Gruppe wird auf dem Januar-Treffen eine aktuelle Knoppix-Linux-FORTH CD erstellen. Über dieses Thema gedenke ich einen Artikel zu verfassen (was ist auf der CD, wie wurde sie erstellt, wie ist die Software zu bedienen, wie erstellt man sich seine eigene CD mit eigenen FORTH-Programmen und Quellen).
Der Artikel wird ca. Anfang Februar fertig werden.

Carsten Strotmann

Betreff: Häresie I
Von: Ulrich Paul <upaul@paul.de>

Wenn man sich die Entwicklung der letzten Jahrzehnte anschaut und dann auf Forth blickt, dann kommt einem der (kalte Angst-)Schweiß: Wir haben in Forth selbst für die primitivsten Operationen immer noch Dutzende von Wörtern, die in modernen Sprachen einfach nichtexistent sind, da der Compiler die Arbeit dem Programmierer abnimmt.

Nein, hier geht es mir nicht um das REORDER, sondern um solche Operationen wie "+,-,!,@", um nur ein paar aufzuzählen. Wenn wir ein Byte irgendwohin speichern wollen, dann schreiben wir c!, bei einem float schreiben wir - ja, was eigentlich? - und bei einem 16-bit-Wert schreiben wir halt nur das Ausrufezeichen. Aber das ist fehleranfällig und unelegant. Also darum:

Forth braucht das "Operator-Overloading"! (Endlich auch.)

Ist das so schwer zu implementieren? Nein, nicht im mindesten. Etwas Aufwand ist es schon, aber ich glaube, der lohnt sich. Mein Vorschlag zur Implementierung:

1.) Der normale Stackkommentar wird Pflicht (das habe ich sowieso schon aus anderen Gründen gefordert). Dabei gelten die gleichen Regeln für Parameternamen, wie sie auch bei anderen Sprachen gelten: Also gleiche Namen bezeichnen gleiche Parameter. Wenn ich schreibe

myfunc (a b c d -- a x y)

dann darf ich innerhalb von myfunc dem Parameter "a" keinen Wert zuweisen. Aber das sollte selbstverständlich sein.

2.) Es gibt einen zusätzlichen "Stackkommentar". Der hat die Form

([int int int int -- int float float]).

(Über die genaue Syntax läßt sich noch reden, aber die o.g. Form hat einen Vorteil: Bestehende Systeme müssen bloß das Wort "([" als einen Alias von "(" definieren und sie können damit Code in der o.g. Form kompilieren, ohne die neuen Features auszunützen. Man kann also schon Code für die Zukunft schreiben - oder einen von einem progressiveren Programmierer nutzen, ohne ihn ändern zu müssen.)

In diesem "Kommentar" - denn eigentlich ist er ja gar keiner mehr - wird der Typ der Parameter angegeben. Logisch, wir müssen uns auf die Bezeichnung der Typen einigen - ich habe halt einfach die genommen, die C verwendet. Aber da sind wir absolut frei. Und man beachte: Ich habe die Klammerung bewußt so gewählt, weil durch eine State-Smart-Implementierung von "({" und "])" das recht leicht implementierbar ist und trotzdem alte Systeme nicht ausgeschlossen werden (s.o.).

Also: Beim Kompilieren wird neben den üblichen Informationen auch noch die "Signatur" eines Wortes abgespeichert. Diese Signatur ist einfach nur die codierte Version ihrer Aufruf- und Rückgabeparameter, bezüglich Anzahl und Typ. Bei einer Verwendung eines solchen Wortes wird überprüft, ob der aktuelle Aufruf sich mit dieser Signatur deckt. Wenn nicht, dann wird weitergesucht, bis ein Wort gleichen Namens, aber der entsprechenden Signatur gefunden wird.

Damit gibt es nur ein Wort namens "+" oder "-" oder ... und der Programmierer muß sich nicht mehr darum kümmern.

So, nur weil ich zu faul bin, die Addition für verschiedene Datentypen explizit in jedem Fall anzugeben, mache ich diesen Aufstand? Nein! Und nochmal nein!

Das wird wichtig, wenn es um Datenein- und Ausgabe geht. Wie wollen wir denn im Internetzeitalter überleben, wenn wir nicht flott und elegant unser Ein/Ausgabeströme umlenken können? Bloß, daß diese Ströme einfach nicht unbedingt forthkompatibel sind, weil sie unterschiedliche Datentypen erwarten.

Klar, wir können für jeden Fall das Problem neu lösen - und jeder für sich (und ohne Synenergieeffekte), aber das ist zum Scheitern verurteilt. Wenn wir so weitermachen, dann kann die FG froh sein, über mehr als ein Dutzend Mitglieder zu verfügen:

1.) Alte sterben weg und keine neue kommen, weil

2.) wir uns lächerlich machen mit unserer Auffassung, daß Forth immer noch - unverändert, wie unsere Fundis es wollen - eine reelle Chance hat. Und das ist schon euphemistisch formuliert. Wir haben wahrscheinlich - alle Forth-Gruppen eingeschlossen - weltweit weniger Mitglieder als die "Flat-Earth-Society" (http://www.flat-earth.org), und das will doch etwas heißen!

Genug gesagt. Entweder da rührt sich was oder das war's.

CU, Uli

Selbstverständlich habe ich die angegebene „Seite“ besucht. Dem Altbergmann Goethe (jawohl, der gehörte zu dieser Zunft) nachempfunden (Faust II), möchte ich sagen: „Ich bin mit Vergnügen da, und freue mich mit diesen; denn von den Burschen kann ich ja auf kluge Köpfe schließen“. ;-)

fep

Betreff: Häresie II
Von: Ulrich Paul <upaul@paul.de>

Hi Fritz,
hier noch eine ketzerische Mail:

Wie wäre es denn, wenn die Forthler missionarisch tätig werden? Sie hergehen und den "armen, verirrten" Programmierern anderer Sprachen die Weihen von Forth lehren? Du lachst? Das geht, aber das Problem ist einfach dieses:

Danach haben andere - weit höher stehende Sprachen - die gleiche Funktionalität wie Forth: und Forth ist damit überflüssig. Aber sie haben zumindest das Forth-Evangelium gepredigt.

Um was geht es:

Forth hat eine sehr nützliche Eigenschaft: Man kann dynamisch zwischen dem Interpreter und dem Compiler umschalten. Das ist äußerst nützlich, wenn man dem Compiler neue Worte beibringen will, während er eigentlich gerade ein Programm compiliert. Ich verwende das leidenschaftlich gern!
Aber das kann man auch in andere Compiler einbauen - in beinahe jeden. Vielleicht sollte sich die Forth-Gemeinde hier einen Grabstein setzen – beizeiten! - bevor sie ohne einen solchen einfach ausstirbt.

Also gut, lassen wir uns an die mageren Kanäle anhängen, die zur Zeit noch Forth am Leben erhalten. Aber wie lange werden diese Kanäle noch offen sein? 5 Jahre? 10 Jahre? Wir dürfen nämlich eines nicht vergessen: Die Rechner werden immer leistungsfähiger und mit meinem Targetcompiler gehe ich nur den Weg, den die Cross-Development-Tools schon seit langem gehen: Entwicklung mit allen Schikanen auf einem leistungsfähigen PC, und das Target ist so klein, daß es nicht einmal das primitivste Forth-System beherbergen könnte. Mit den neuen Schnittstellen gibt es absolut keine Notwendigkeit mehr Forth zu implementieren - weder zum Debuggen, noch zu erweiterten Funktionstests. Also in der Richtung ist Forth einfach tot!

Was wir also machen können, ist, unser Erbe weiterzugeben und die Inschrift auf unserem Grabstein beizeiten positiv selbst zu gestalten – oder endlich und radikal umdenken. Aber "umdenken" ist genauso ketzerisch, wie wenn ich fordern wollte, daß der Papst seinen Unfehlbarkeitsanspruch aufgeben sollte.

Soviel zu diesem Thema!

CU, *Uli*

Eine erste Antwort möchte ich gleich selbst wagen. Ich betreue hauptberuflich zur Zeit unter anderem ein von mir initiiertes Projekt, bei dem es darum geht, eine unserer größeren Maschinen soweit zu automatisieren, daß eine bestimmte Menge der zur Verfügung stehenden Funktionen dem Maschinenführer „vorgelegt“ wird. Der Mann an der Maschine soll die Verantwortung für das Arbeitsergebnis der Maschine aus mehreren Gründen vollständig behalten. Aber ein Rechner soll ihn von allen unkritischen Routinearbeiten entlasten und ihn darin unterstützen, sich auf die eigentlichen Entscheidungsprozesse seiner Arbeitsaufgabe zu konzentrieren. Das erfordert unter anderem, daß in einer Anzahl unterschiedlicher Voraussetzungen in situ eine Parametrierung bei laufender Maschine notwendig ist.

Alle notwendige Hardware, einschließlich der Sensorik, einschließlich des Rechners und natürlich einschließlich der Maschine selbst steht zur Verfügung. Es gibt aber kein rechtes Vorankommen, weil die Programmierer, die diese Aufgabe übernehmen sollen, in SPS denken, mit C++ programmieren, einen 68000 vergewaltigen und schlicht nicht glauben wollen, daß es so etwas wie Forth auch nur als Konzept gibt. Ich habe versucht, ihnen HOLON zu erklären. Ich werde es ihnen zeigen müssen. Vermutlich glauben die das dann trotzdem nicht.

Ich beginne zu verstehen, daß das eigentliche Problem viel tiefer liegt. Maschinenbauer können (und wollen) nicht programmieren. Elektriker programmieren ausschließlich SPS. Informatiker beschäftigen sich entweder mit mathematischen Problemen, oder mit lukrativer Software für die Börse oder geschäftliche Transaktionen oder mit der Administration von Netzwerkfeatures die niemand wirklich braucht.

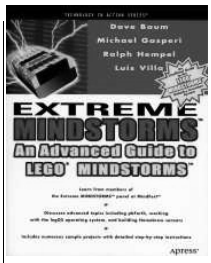
Wir sagen, daß Forth einfach und universell sei. Ich denke, daß es das auch ist. Aber Forth ist dann einfach zu lernen, zu lehren und letztlich auch zu handhaben, wenn man sich praktischen Problemen aus der Praxis heraus nähert, wenn man “das Programmieren“ sozusagen von der Pike auf gelernt hat. Solange der Praktiker dem (programmierenden) Spezialisten sagt: „Ich möchte doch nur...“, und zur Antwort bekommt: „... aber man könnte doch...“, solange wird Forth einfach deshalb Akzeptanzprobleme haben, weil es zu einfach und zu vernünftig ist. *fep*

Von: Wolf Wejgaard <wejgaard@holonforth.com>

...

Schöne Arbeit, Fritz. Da läuft ja schon mal was. S-Records sind mir willkommen, Holon11 produziert bereits wahlweise Codefiles im S-REC Format (S1 und S9). Motorola verwendet gerne 32 Bytes pro Records, da dann ein Record gerade eine Zeile belegt als Text - denke ich.

...



(K)ein neues Forth-Buch, aber trotzdem ein Buch, das in meiner Sammlung von Forth-Büchern nicht fehlen durfte, ist „Extreme Mindstorms“ von Baum, Gasperi, Hempel und Villa. „An Advanced Guide To Lego Mindstorms“ gibt eine brauchbare, wenn auch keine wirklich „advanced“ Einführung in den RCX, den „Robotorbau“, die Firmware 2.0 von LEGO, und vieles mehr. Immerhin ist darunter auch eine Einführung in Ralf Hempels pbForth zu finden, sowie ein Programmierbeispiel in pbForth für einen Roboter und ein Anhang mit Forth-Worten, die nach Ansicht der Autoren besonders gebräuchlich sind.

Mindestens der gleiche Raum, den pbForth in diesem Buch einnimmt, wird auch den beiden anderen, alternativen Entwicklungsumgebungen NQC (NotQuiteC) und LegOS eingeräumt. Für Bastler von besonderem Interesse werden die Kapitel zu selbstgefertigten aktiven und passiven Sensoren sein. Auf rund 330 Seiten hilft „Extreme Mindstorms“ bei dem Einstieg in eine intensivere Beschäftigung mit dem RCX.

Leider wird das Buch nur in englischer Sprache angeboten. Das erschwert sicher manchem den angebotenen Einstieg. Zusätzlich wird die Kaufentscheidung durch den nicht gerade jugendgerechten Preis von 29,95 $ US erschwert.

Zu Bestellen sind die extremen Geistesblitze beim APRESS Verlag unter der ISBN 1-893115-84-4.

fep

Die im Web angebotene Version von Holon86 ist vollkommen frei (Freeware), auch für Entwicklungsarbeiten, falls sich das jemand antun will. Wie du siehst, ist der Umfang an Utilities beschränkt.

Wer aber echt mit Holon86 arbeiten will, kriegt von mir die guten Sachen gratis. Mail genügt.

Vielleicht biete ich später wieder eine volle Profiversion mit Versionskontrolle und Validierung und Unterstützung.

...

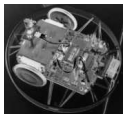
Wolf

Der Ausschnitt aus einer Mail Wolf Wejgaards entstammt recht intensiver Korrespondenz, die Wolf, Martin Bitter und ich bezüglich Martins und meiner Feiertagsbeschäftigung mit HOLON geführt haben. Die Information, daß HOLON86 ohne jegliche Einschränkung frei verfügbar ist, veranlaßt vielleicht den einen oder anderen Leser, sich diese forthige Entwicklungsumgebung doch noch einmal genauer anzusehen. Herunterladen können Sie HOLON (und mehr) von

http://www.holonforth.com

Das HOLON86 ist in englischer Sprache dokumentiert. Für die frühere „Schnupperversion“ HOLON4th habe ich alle Dokumentationstexte ins Deutsche übersetzt. Diese Übersetzung läßt sich problemlos für HOLON86 nutzen. Bei Bedarf bitte ich, diese Texte per Mail bei mir anzufordern.

fep



USHI-Tag in Utrecht

Bericht vom

USHI-Tag

in Maarssen
(Utrecht)

Zum Samstag, den 10. Jan. 2004, hatten unsere niederländischen Kollegen zum Ushi-Tag nach Maarssen eingeladen. Maarssen ist eine kleine Stadt am Westrand von Utrecht und von Hamminkeln aus in cirka 1,5 Stunden über die Autobahn leicht zu erreichen. Nachdem Martin Bitter es fertig gebracht hatte, mich für einen Besuch in Maarssen zu begeistern, habe ich mich an diesem Samstagmorgen tatsächlich von meinen eigenen forthigen Vorhaben losgerissen, und mit Martin Bitter eben diesen Ushi-Tag besucht.

Ushi sollte Ihnen kein Unbekanter mehr sein. Die Teilnehmer der Forthtagung in Garmisch-Partenkirchen in 2002, werden sich an die Ushis erinnern, die Willem Ouwerkerk und Albert Nijhof trotz Willems damals akuter, schwerer Erkrankung dort vorgestellt haben. Einen ersten Bericht über die Ushis konnten Sie in der VD lesen (03/2002).

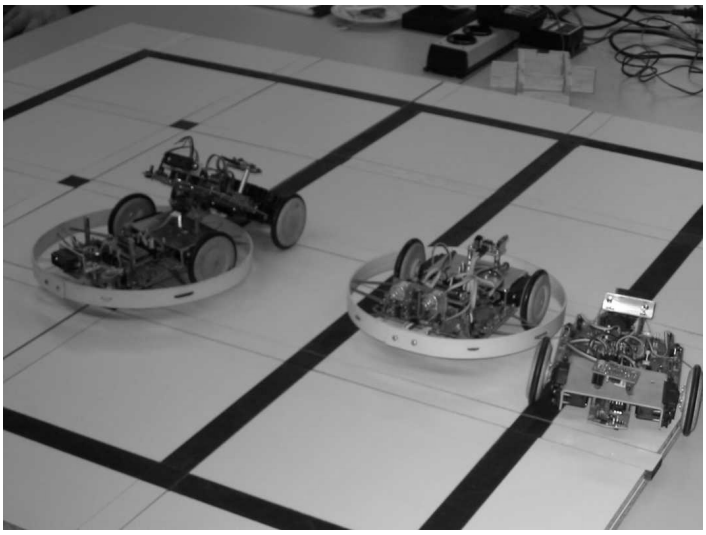
Der Ushi-Tag fand in Maarssen anläßlich des ersten, turnusmäßigen Treffens der niederländischen Forther im neuen Jahr statt. Von elf bis drei Uhr am Nachmittag sollte das Treffen dauern, das Willem Ouwerkerk mit einem Vortrag über die Hardware der Ushis und über das AVR-ByteForth eröffnet hat. Zu meinem Glück verstand Martin ausreichend holländisch, um mir zu übersetzen, was ich „nit snappen“ konnte.

14 Forther aus den ganzen Niederlanden, später waren es 15, die sich monatlich treffen, hatten ganz offensichtlich großen Spaß mit ihren Ushis. Auf einem frei konfigurierbaren Labyrinth konnten die kleinen Roboter Wege abfahren, oder miteinander „rangeln“ und sich gegenseitig aus dem Spielfeld drängen. Teilweise waren 4 Ushis gleichzeitig „im Ring“, drängelten, kletterten aufeinander, schoben und hoben und ließen durchaus den Eindruck entstehen, daß etwas Lebendiges zu beobachten wäre. Martins hat eine spontane Idee glücklicherweise nur in deutscher Sprache und auch nur leise artikuliert – man könnte doch im nächsten Jahr mit dem RCX... Aber da schauen wir erst noch einmal genauer hin. Vielleicht finden sich interessierte Mitstreiter und Begleiter.

Die niederländischen Forther scheinen mir Bastler im positivsten Sinn des Wortes zu sein. Die Diskussion mit Willem um die Möglichkeiten, mit dem Ushi Forth zu lernen und zu lehren, wurden, wenn ich das richtig „gesnappt“ habe, mit Bemerkungen kommentiert, aus denen ich entnommen habe, daß zu den Ushis eher ein großes Potential zum Lernen und Lehren im Umgang mit der elektronischen Hardware gesehen wird. In diesem Zusammenhang ist dann nicht mehr verwunderlich, wenn Marc Hartjes einen Fehler in der Sensorik und der Steuerung seines Ushis nicht im Programm gesucht hat, sondern lieber gleich das Oszilloskop auspackte und mit Martin eine Diskussion über die Widerstände in NiCd Akkus, Piks mit einer Frequenz von 5 Mikrosekunden, und den richtigen Einsatz von Spannungsteilern angestrengt hat.

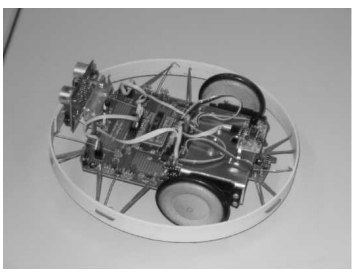
Anstrengend war das Treffen für mich, weil ich die Sprache nicht verstehe. Am Ende war ich recht froh, als Martin sich bereit erklärte, die Heimreise bereits irgendwann zwischen eins und zwei Uhr am frühen Nachmittag anzutreten. Von Willems AVR-ByteForth habe ich darum gar nichts mehr sehen können. Vielleicht zeigt er mir das auf Fehmarn.

fep



Vier Ushis rangeln um den Platz im Labyrinth.

Diskussion zur Ushi-Hardware.



Ushi-Bausätze und das AVR-ByteForth können von Willem Ouwerkerk käuflich erworben werden.

W.ouwerkerk@kader.hobby.nl

Eine in Forth programmierte Lösung zum Viererproblem

Ewald Rieger
ewald.rieger@t-online.de
Littersheimer Weg 10
D-67240 Bobenheim-Roxheim

22.03.2003

Zusammenfassung

Beim Viererproblem sollen die natürlichen Zahlen von 1 bis 50 unter Anwendung der Rechenoperatoren (+ - * / sqrt() ()² !n) mit genau viermal der Ziffer Vier berechnet werden. Die gestellten Teilaufgaben verlangen die Operatoren in unterschiedlichen Kombinationen einzusetzen. Eine weitere Aufgabe besteht darin, die maximale Anzahl Lösungsvarianten und die Variante mit der kleinsten Anzahl an Operatoren zu jeder Zahl zu berechnen. Dieser enorme Rechenaufwand kann nur durch ein Computerprogramm in vernünftiger Zeit gelöst werden. Das nachfolgende Programm zeigt, wie man das Viererproblem mit Hilfe von Forth lösen kann.

1. Motivation

Das Rätsel zum Viererproblem erschien in der VD 2/2000. Zwei Ausgaben später veröffentlichte die VD zwei Teillösungen zum Viererproblem. Beide Lösungen entstanden durch Knobeln mit Papier und Bleistift. In einem Fall wurden die Ergebnisse wenigsten mit Hilfe von Forth überprüft. Auch wenn Fred Behringer freistellte, wie man diese Aufgabe lösen möchte, so kann es doch kein Zufall sein, daß dieses Rätsel ausgerechnet in der VD, einem Magazin der Computersprache Forth, erschienen ist. So regten mich die beiden Rätselauflösungen, deren Fleiß ich in keiner Weise schmälern möchte, wenigstens dazu an, das Problem durch ein Forth-Programm zulösen.

2. Aufgabenstellung

1. Man drücke die natürlichen Zahlen von 1 bis 50 dadurch aus, daß man die Ziffer "vier" genau viermal verwendet und keine andere als die folgenden Rechenoperationen zuläßt: +, -, /, *, die Fakultät (!n = 1*2*3*n), die Quadratwurzel SQRT() und das Quadrat ()².

2. Kann man mit + - / * und der Quadratwurzel auskommen?

3. Gibt es andere Kombinationen von zulässigen Operationen, mit denen es auch geht?

4. Wie sieht die kleinste Zahl von Operationen, mit denen es auch geht, aus?

5. Gibt es mehrere Operationskombinationen, deren Zahl die kleinste Zahl nicht überschreitet?

6. Fallen Ihnen weitere Fragen ein?

7. Man gebe die größtmögliche Zahl von Lösungsvarianten für (1) an, indem man die einzelnen Varianten unmittelbar benennt.

3. Überlegungen zum Algorithmus

3.1 Datenrepräsentation

Zur Lösung der gestellten Aufgaben empfiehlt sich eine Entwicklung der Zahlen in einem DAG (Direkt Acyclischer Graph). Wir beginnen in den Blättern des Baumes mit den Vierern und verknüpfen sie durch Anwenden einer Rechenoperation zu einem Knoten, der eine neue Zahl repräsentiert. Für die Rechenoperationen + - / * benötigen wir zwei mal die Vier, während die Operationen !n, Sqrt(), und das Quadrat nur eine Vier verbrauchen. Jeder neu berechnete Knoten zeigt auf seine (n) Vorgänger und merkt sich die angewandte Rechenoperation. Alle zugelassenen Rechenoperationen werden durchlaufen und die Knoten in der Liste "Neue" gespeichert. Zur späteren Bewertung berechnen wir für jede Zahl zusätzlich den Verbrauch an Vierern und die Anzahl von Rechenoperationen gleich mit. Beide Werte sind in ihren Zahlenknoten zusätzlich vermerkt. Nach einer Rechenrunde, in der alle anzuwendenden Rechenoperationen jeweils einmal mit den schon gefundenen Knoten verknüpft wurden, vereinigt man die Liste *Neue* mit der Ergebnisliste in der Weise, daß ein
Knoten mit einer gewünschten Eigenschaft nur einmal in der Ergebnisliste vorhanden ist. Zur Lösung der Teilaufgaben 1, 2, 3 und 5 genügt es, den Vergleich auf die Eigenschaft: errechnete Zahl und den Verbrauch an Vierern; zu beschränken. Bei den Teilaufgaben 4 und 7 sind die Kosten an Operationen, die zu einer Zahl führten, mit einzubeziehen. Weiter ist die Kommutativität der Operationen + und * zu berücksichtigen. In jeder weiteren Rechenrunde werden alle Zahlenknoten der vorausgegangenen Runden mit allen Rechenoperationen erneut kombiniert, in der Liste "Neue" gespeichert und wie oben beschrieben mit der Liste "Ergebnis" vereinigt. Dieser Vorgang wird solange wiederholt bis keine neuen Zahlenknoten unter Beachtung der weiter unten beschriebenen Bedingungen entstehen und sich zur Ergebnisliste hinzufügen lassen.

3.2 Terminierung

Neu berechnete Zahlenknoten werden nur berücksichtigt, wenn sie zwischen der kleinsten und der größten zu berechnenden Zahl liegen und der Verbrauch an Vierern die Anzahl von vier nicht übersteigt.

3.3 Produktivität der Rechenoperatoren

Unter einigen Umständen liefern manche Operationen keine neuen Zahlenwerte oder bilden Schleifen und könnten so zu unendlich vielen Wegen beitragen. So ist es nicht sinnvoll nachfolgende Operationen auszuführen:

1. die Fakultät von 1 und 2

2. die Wurzel von 1

3. das Quadrat von 1

4. das Anwenden von ()² und sqrt() in direkter Folge und umgekehrt.

5. sqrt(), wenn das Ergebnis keine natürliche Zahl ist.

6. die Division, wenn sie keine natürliche Zahl liefert.

4. Werkzeuge

Die Programmierung erfolgte in Forth mit Erweiterungen zur objektorientierten Programmierung. Zusätzlich wurde ein Tool zur dynamischen Verwaltung von Objekten in Listen eingesetzt. Ihre Organisationsformen erstrecken sich auf LIFO, FIFO und sortierte Listen, die bei Bedarf beliebig tief geschachtelt werden können. Die Methode insert einer Liste fügt ein neues Objekt so ein, daß die gewünschte Organisationsform erhalten bleibt. Das Extrahieren von einzelnen Objekten oder Listen von Objekten mit bestimmten Eigenschaften wird unterstützt. Zur Zeit ist das Programm auf Bigforth unter Linux, Windows und DOS lauffähig. Zur Erprobung der Algorithmen zeigte sich die Interaktive und inkrementelle Anwendung der Tools als sehr hilfreich. Objekte verbesserten die Lesbarkeit der Quelltexte und schufen Transparenz in komplexen Datenstrukturen.

5. Programmierung

Der nachfolgende Quelltext beschreibt den wesentlichen Programmteil zum Viererproblem. Auf die Beschreibung der Tools wurde aus Übersichtlichkeit an dieser Stelle verzichtet.

5.1 Rechenoperatoren

```
: q ( n -- n )
    dup 1 = IF drop 100000 EXIT THEN dup * ;

: f ( n -- n )
    dup 3 < IF drop 100000 EXIT THEN 1 swap
    0 ?DO I 1+ * LOOP ;

: w ( n -- n )
    100000 swap 100 2 DO I dup * over =
    IF drop I swap leave THEN LOOP drop ;

: div
    dup 0= IF 2drop 1000000 EXIT THEN
    /mod swap IF drop 1000000 THEN ;
```

Die Operatoren Quadrat "q" , Squart() "w" und die Fakultät "f" sind so definiert, daß unproduktive Operationen einen Wert außerhalb des gültigen Zahlenbereichs zurückgeben, bzw. nur gültige natürliche Zahlen als Resultat erzeugen. Die Forth-Division "/" wird durch "div" ersetzt. Durch diese Maßnahmen wird eine Terminierung bei unzulässigen Werten erwirkt. Alle anderen Operatoren (*, +, -) kommen in ihrer Standarddefinition zur Anwendung.

```
s" +" string: : $+ $+ +ref
s" *" string: : $* $* +ref
s" q" string: : $q $q +ref
s" w" string: : $w $w +ref
s" f" string: : $f $f +ref
s" /" string: : $/ $/ +ref
s" -" string: : $- $- +ref
```

Die Operatoren werden in den Zahlenknoten durch eine Referenz auf einen dieser Strings repräsentiert.

5.2 Der Zahlenknoten

5.2.1 Die Objektschnittstelle

```
data: class zahlknoten
method zahl>          ( -- z )
method >zahl          ( z -- )
method verbrauch>     ( -- v )
method >verbrauch     ( v -- )
method verbrauch-     ( cfa o -- flg )
method gleich1-       ( cfa o -- flg )
method gleicha-       ( cfa o -- flg )
method w?             ( -- flg )
method q?             ( -- flg )
method count-ops      ( -- n )
method operationen>   ( -- ops )
method >operationen   ( ops -- )
method operationen-   ( cfa o -- flg )
method fetch          ( -- z v ops )
method store          ( z v ops -- )
```

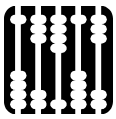
Diese Methoden bilden die Schnittstelle der Klasse Zahlenknoten. Hinzu kommen die geerbten Methoden der Klasse Data: zur Verwaltung der Objekte in Knoten und den davon abgeleiteten Listen. Zur Konvention der Methodennamen gilt:

- *>method* ist eine Methode, die Parameter entsprechend ihrem Stackdiagramm in das Objekt schreibt.

- *method>* ist eine Methode, die Parameter entsprechend ihrem Stackdiagramm aus dem Objekt auf den Datenstack schreibt.

- *method-* verlangt ein Objekt gleicher Klasse als oberstes Stackelement zum Vergleich. Die Methode verhält sich ähnlich dem Forth-Wort "-text". Ist Flg=0 sind beide gleich, flg<0 zeigt einen kleineren Wert im Vergleichsobjekt und flg>0 einen grösseren Wert an. Auf dieses Flag wird die cfa (von 0=, 0>, 0<, 0<> oder noop) angewendet. Dies wandelt das Flag in ein Abbruchskriterium einer Iteration über eine Liste um. Die Hauptanwendung liegt beim Einfügen in sortierte Listen und Extrahieren von Objekten mit bestimmten Eigenschaften aus einer Liste.

Die Methode *w?* prüft seinen direkten Vorgänger auf den Operator sqrt(). Flg=0 bedeutet, daß sein Vorgänger nicht durch sqrt() gebildet wurde. Sinngemäß prüft q? auf das Quadrieren seines direkten Vorgängers. Die Methode *gleich1-* kommt zur Anwendung, wenn eine Liste berechnet wird, welche nur eine Variante zu einer Zahl enthalten soll. Die Methode *gleicha-*



vergleicht in der Weise, daß alle Wege zu einer Zahl berechenbar sind. *Count-ops* erlaubt die nachträgliche Zählung der Operatoren durch eine Tiefensuche, die zu einer Zahl führten.

5.2.2 Die Exemplarvariablen

```
spnode: ptr vorgaenger
cell Var Zahl
cell Var Verbrauch
cell Var operationen
```

Die Objektvariable *Zahl* enthält die natürliche Zahl vier oder eine Zahl, welche über vorausgegangene Operationen berechnet wurde. Die Variable *Verbrauch* zählt dabei den Verbrauch an Vierern, während die Variable *operationen* die Operatoren summiert, die zu dieser Zahl führten. *Vorgaenger* ist ein Knoten, der Zeiger auf eine oder zwei vorausgegangene Zahlenknoten und einen Zeiger auf einen String einer Operation aufnimmt.

5.2.3 Methodenimplementierung

```
how:
: zahl> zahl @ ;
: >zahl zahl ! ;
: >verbrauch verbrauch ! ;
: verbrauch> verbrauch @ ;
: print (( einruecken @ 3 < IF .cr THEN
     einruecken @ 3 < IF zahl> . '= emit space
                    THEN
     vorgaenger len 0= IF zahl> . THEN
     vorgaenger len IF vorgaenger print
                  THEN )) ;
```

Das *Wort* ((in *print* zählt die Verschachtelungstiefe der Zahlenknoten in der Variablen *einrücken* mit. Die Variable *einrükken* entscheidet, ob wir Zahlen links oder rechts des Gleichheitszeichen drucken. Ist kein Vorgaenger vorhanden, wird die Zahl des Knotens gedruckt (nur bei vier), sonst wird in seine Vorgaenger hinabgestiegen und gedruckt.

```
: >operationen ( n -- ) operationen ! ;
: operationen> ( -- n ) operationen @ ;
: operationen- ['] noop over class-
     IF >o operationen> o> operationen> - swap
           execute
     ELSE 2drop false THEN ;
: fetch ( -- z v ops ) zahl> verbrauch>
     operationen> ;
: store ( z v ops -- ) >operationen >verbrauch
     >zahl ;
: verbrauch- ( cfa o -- flg )
     ['] noop over class-
     IF >o verbrauch> o> verbrauch> - swap
        execute
     ELSE 2drop false THEN ;
```

Verbrauch- wird an dieser Stelle beispielhaft für andere Vergleiche erklärt. Es vergleicht den Verbrauch an Vierern zwischen zwei Zahlenknoten. Zunächst überprüft class- die Klassenindentität. Bei ungleichen Klassen wird der Vergleich abgebrochen und false zurückgegeben. Andernfalls wird der Verbrauch von beiden Objekten auf den Datenstack gelegt und die Differenz gebildet. Anschließend führt execute die cfa von 0<,0>,0<> oder 0= aus. Diese Eigenschaft der Flagbildung ermöglicht, Listen leicht zu sortieren oder Objekte mit bestimmten Eigenschaften aus Listen zu extrahieren.

```
: gleich1-
     ['] noop over class-
     IF >o zahl> verbrauch> o> zahl>
        verbrauch>
     rot = -rot = and swap execute ELSE 2drop
     false THEN ;

: init ( z v op -- )
     super init >operationen >verbrauch >zahl
     spnode: new bind vorgaenger ;
```

Init wird innerhalb der Methode *new* aufgerufen und initialisiert das neue Objekt mit *Zahl, Verbrauch* und *Operantenverbrauch;* die Werte werden vom Datenstack genommen. Zusätzlich wird ein neuer leerer Knoten an *Vorgaenger* gebunden.

```
: dispose ref @ ?EXIT vorgaenger dispose super
    dispose ;
```

Dispose löscht ein Objekt und gibt seinen Speicher an den Heap zurück. Ein Objekt darf aber erst gelöscht werden, wenn es von keiner anderen Liste und keinem anderen Objekten referenziert wird.

```
: gleicha- ( cfa o -- flg )
     ['] noop over class-
     over >o vorgaenger len o>
     vorgaenger len - 0= or dup
     IF swap >r
        vorgaenger len 3 = IF
        r@ >o 0 vorgaenger tes o>
              0 vorgaenger tes = and
        r@ >o 1 vorgaenger tes o>
              1 vorgaenger tes = and
        r@ >o 2 vorgaenger tes o>
              2 vorgaenger tes = and THEN
              vorgaenger len 2 =
        IF  r@ >o 1 vorgaenger tes o>
                  1 vorgaenger tes = and
            r@ >o 0 vorgaenger tes o>
                  0 vorgaenger tes = and THEN
     rdrop swap execute
     ELSE 2drop false THEN ;
```

Gleich- vergleicht paarweise die Zeiger der Vorgaengerobjekte. Diese Methode wird zur Berechnung der Aufgabe 7 benötigt.

```
: w? $w self vorgaenger find-one ;
: q? $q self vorgaenger find-one ;
: value-
     ['] noop over class-
     IF >o zahl> maxoperationen @ *
        operationen> + o>
           zahl> maxoperationen @ *
        operationen> +
        - swap execute
     ELSE 2drop false THEN ;
```


```
 : count-ops ( n -- n+1 )
     vorgaenger data self 0= ?EXIT
     vorgaenger self node: with
     :[ ^ node: with data self string: with &
     string: @ class?
        IF 1+ THEN endwith
        data self zahlknoten with & zahlknoten
        @ class?
        IF count-ops THEN endwith endwith
     ]: data all endwith ;
class;
```

5.3 Listen zur Verwaltung der Daten und spezialisierte Operatoren zu diesen Listen

5.3.1 Die Ergebnisliste

```
sort: : ergebnis
```

Diese Liste enthält alle Zahlenknoten, die als Zwischen- und Endergebnisse während der Entwicklung des Zahlenbaumes entstehen.

```
4  1 0 zahlknoten : vier vier +ref
44 2 0 zahlknoten : 44   44   +ref
```

Von den Knoten *vier* und *44* leiten sich alle weiteren Zahlenknoten ab (keine Redundanz erlaubt).

```
0 4 0 zahlknoten : 4verbrauchmuster
```

Das *4Verbrauchmuster* erlaubt in den nachfolgend definierten Worten, Zahlenknoten mit den Endeigenschaften oder aber weiterzuentwickelnde Zahlenknoten aus der Liste "Ergebnis" in eine andere Liste zu extrahieren.

```
: alle-ende ( node -- node )
     4verbrauchmuster  '  verbrauch- ergebnis
     property !
                       ['] 0= ergebnis
     relation !
     4verbrauchmuster self swap ergebnis
     find-all ;
: alle-4kleiner ( node -- node )
     4verbrauchmuster  '  verbrauch- ergebnis
     property !
                       ['] 0< ergebnis
     relation !
     4verbrauchmuster self swap ergebnis
     find-all ;

Variable vergleich zahlknoten
         ' gleich1- vergleich !

: in-ergebnis-enthalten? ( o1 -- o2/false )
     vergleich @ ergebnis property !
     ['] noop ergebnis relation !
     ergebnis find-one ;
```

In-ergebnis-enthalten? prüft die Liste "Ergebnis", ob ein Objekt mit gleichen Eigenschaften entsprechend dem Vergleichsobjekt enthalten ist und gibt bei Identität seinen Objektpointer zurück, sonst false. Die Cfa in der Variablen *Vergleich* entscheidet, welche Vergleichsmethode angewendet wird.

5.3.2 Weitere Listen

```
sort: : neue
sort: : hilfe
```

5.3.3 Einige Tests

```
1 4 0 zahlknoten new ergebnis insert
2 3 0 zahlknoten new ergebnis insert
3 2 0 zahlknoten new ergebnis insert
4 1 0 zahlknoten new ergebnis insert

ergebnis print neue empty neue self alle-ende
drop

cr .( ende ) neue print

neue empty neue self alle-4kleiner drop

cr .( 4< ) neue print
```

5.4 Konstruktion des Zahlenbaums

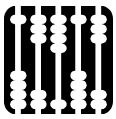
```
: ?zahlknoten ( z v ops -- node/0 )
     >r over kleinste-zahl @ groesste-zahl @
     within
        over groesster-verbrauch @ < and
     IF r> zahlknoten new ELSE rdrop 2drop
     false THEN ;
```

?zahlenknoten nimmt die Parameter für einen neuen Zahlenknoten vom Datenstack, erzeugt ihn und gibt den Objektpointer des neuen Zahlenknoten zurück. Sind die Werte außerhalb des gültigen Bereiches wird false zurückgegeben.

```
: merke3 ( o1 o2 str-o zk/0 -- zk/0 )
     ?dup IF >r
     $* self over = >r $+ self over = r> or
     IF >r 2dup > IF swap THEN r> THEN
     r> zahlknoten with vorgaenger self
     node: with swap insert insert insert
           endwith
     self endwith
     ELSE 2drop drop false THEN ;
```

Merke3 erwartet zwei Objektpointer der Vorgaenger o1 und o2, aus denen zk gebildet wurde, sowie einen Objektpointer auf einen String der Operatoren +, -, *, / und fügt diese Pointer in den Vorgaengerknoten von zk ein. Ist zk=0, so handelt es sich um einen leeren Pointer. In diesem Falle werden die Parameter verworfen und 0 zurückgegeben.

```
: merke2 ( o1 str-o zk/0 -- zk/0 )
   ?dup IF >r
   dup $q self =
   IF over zahlknoten with w? endwith ELSE 0 THEN >r
   dup $w self =
   IF over zahlknoten with q? endwith ELSE 0 THEN r> or
   IF r> zahlknoten with dispose endwith 2drop false
     EXIT THEN
   r> zahlknoten with vorgaenger self
   node: with swap insert insert endwith self endwith
   ELSE 2drop false THEN ;
```



Holländisch ist gar nicht so schwer. Es ähnelt sehr den norddeutschen Sprachgepflogenheiten. Und außerdem ist Forth sowieso international. Neugierig? Werden Sie Förderer der

HCC-Forth-gebruikersgroep.

Für 10 € pro Jahr schicken wir Ihnen 5 oder 6 Hefte unserer Vereinszeitschrift ‘Het Vijgeblaadje’ zu. Dort können Sie sich über die Aktivitäten unserer Mitglieder, über neue Hard- und Softwareprojekte, über Produkte zu günstigen Bezugspreisen, über Literatur aus unserer Forth-Bibliothek und vieles mehr aus erster Hand unterrichten. Auskünfte erteilt:

Willem Ouwerkerk
Boulevard Heuvelink 126
NL-6828 KW Arnhem
E-Mail: ***w.ouwerkerk@kader.hobby.nl***

Oder überweisen Sie einfach 10 € auf das Konto 525 35 72 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden Willem Ouwerkerk zu wenden.

Merke2 arbeitet sinngemäß wie *merke3* für die Operatoren f, w, q, die nur einen Zahlenknoten als Vorgaenger haben.

```
Create 0verbraucher
' w , $w self ,
' q , $q self ,
' f , $f self ,
DOES> ( o1 nr -- ) 2dup >r >r swap 2*
cells + cell+ @ over zahlknoten with
fetch endwith rot
r> r> swap 2* cells + perform -rot 1+
?zahlknoten merke2 ?dup IF neue
insert THEN ;
```

0Verbraucher nimmt den Pointer eines Zahlenknotens und den Index für eine Operation vom Datenstack und führt diese Operation aus. Erzeugt danach einen neuen gültigen Zahlenknoten und fügt ihn in die Liste "neue" ein.

```
Create +verbraucher
' + , $+ self ,
' - , $- self ,
' * , $* self ,
' div , $/ self ,
DOES> ( o1 o2 nr -- ) 2dup >r >r swap
2* cells + cell+ @
2 pick zahlknoten with fetch
endwith >r
3 pick zahlknoten with fetch
endwith >r
rot + -rot r> r> + -rot r> r> swap 2*
cells + perform -rot
1+ ?zahlknoten merke3 ?dup IF neue insert
THEN ;
```

+Verbraucher erzeugt sinngemäß zu *0Verbraucher* neue Zahlenknoten für die Operatoren +, -, *, /.

```
: alle-0verbraucher ( o1 -- )
     3 0 DO dup I 0verbraucher LOOP drop ;
: alle0 ( -- )
    :[ ^ node: with data self endwith
     alle-0verbraucher ]:
    ergebnis data all ;
```

Alle0 iteriert über alle Zahlenknoten in der Liste "Ergebnis" und fügt neue Knoten in die Liste "neue" ein.

```
: alle-+verbraucher ( o1 o2 -- )
    4 0 DO 2dup I +verbraucher LOOP 2drop ;
: alle+ ( o1 -- )
    :[ dup ^ node: with data self endwith
     alle-+verbraucher ]:
     ergebnis data all drop ;
```

Alle+ nimmt den Pointer eines Zahlenknoten vom Datenstack und kombiniert diesen mit allen Zahlenknoten der Liste "Ergebnis" durch.

```
: vereinigen
    BEGIN neue remove ?dup
    WHILE dup in-ergebnis-enthalten?
      IF zahlknoten with dispose endwith
      ELSE
      zahlknoten ' value- ergebnis property !
      ['] 0> ergebnis relation !
      ergebnis insert THEN
    REPEAT ;
```

Nach einem Durchlauf werden Zahlenknoten aus der Liste "Neue" entnommen. Ist ein Knoten entsprechend der angewandten Vergleichsmethode schon in der Liste "Ergebnis" enthalten wird dieser Knoten gelöscht, sonst in die Liste "Ergebnis" eingefügt.

```
: step
    neue empty neue self alle-4kleiner drop
    alle0 BEGIN neue len dup ?cr .
          WHILE neue remove alle+ REPEAT
    vereinigen ;
```

Schließlich führt "step" eine komplette Runde aller Operatoren über die Liste "Ergebnis" aus.

5.5 Ein Rechenbeispiel

```
zahlknoten ' gleich1- vergleich !
 1 kleinste-zahl !
 5 groesste-zahl !
50 groesster-verbrauch !
 0 iterationen !
ergebnis empty
vier self ergebnis insert
44 self ergebnis insert
neue empty
[BEGIN] ergebnis len iterationen ! step
ergebnis len iterationen @ = [UNTIL]
hilfe empty zahlknoten
' value- hilfe property !
' 0> hilfe relation !
hilfe self alle-ende drop
cr hilfe print cr hilfe len .
```

Dieses Beispiel berechnet Lösungsvarianten der Zahlen von 1 bis 50. Wird *gleich1-* durch *gleicha-* ersetzt, werden alle Varianten für den gegebenen Zahlenbereich berechnet. Zu Beginn wird die Liste Ergebnis mit dem Startknoten *vier* und *44* vorbelegt. Dann wird Schicht um Schicht der Baum aufgebaut, bis keine neuen Knoten entstehen. Zum Schluß extrahieren wir die gewünschte Endkonstellation aus der Ergebnisliste in die Liste "Hilfe" und drucken sie aus.

6. Ergebnisse

6.1 Eine Lösung zu Aufgabe (1)

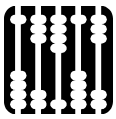
Die Zahlen 1 bis 50 lassen sich lückenlos berechnen.

```
 1 = ( ( 4 * 4 ) / ( 4 * 4 ) )
 2 = ( ( 4 * 4 ) / ( 4 + 4 ) )
 3 = ( ( 4 * ( f 4 ) ) / ( ( q 4 ) + ( q 4 ) ) )
 4 = ( ( 4 * ( 4 * 4 ) ) / ( q 4 ) )
 5 = ( ( ( q 4 ) + ( f 4 ) ) / ( 4 + 4 ) )
 6 = ( ( 4 * ( f 4 ) ) / ( 4 * 4 ) )
 7 = ( ( 4 + 4 ) - ( 4 / 4 ) )
 8 = ( ( 4 * 4 ) - ( 4 + 4 ) )
 9 = ( ( 4 + 4 ) + ( 4 / 4 ) )
10 = ( ( ( 4 * 4 ) + ( f 4 ) ) / 4 )
11 = ( ( ( f 4 ) - ( w 4 ) ) / ( 4 / ( w 4 ) ) )
12 = ( ( ( f 4 ) * ( 4 + 4 ) ) / ( q 4 ) )
13 = ( ( ( f 4 ) + ( w 4 ) ) / ( 4 / ( w 4 ) ) )
14 = ( ( ( q 4 ) + ( f 4 ) ) - ( ( f 4 ) + ( w 4 ) ) )
15 = ( ( 4 * 4 ) - ( 4 / 4 ) )
16 = ( ( 4 * 4 ) / ( 4 / 4 ) )
17 = ( ( 4 * 4 ) + ( 4 / 4 ) )
18 = ( ( ( q 4 ) + ( f 4 ) ) - ( ( f 4 ) - ( w 4 ) ) )
19 = ( ( f 4 ) - ( 4 + ( 4 / 4 ) ) )
20 = ( ( ( f 4 ) + ( f 4 ) ) - ( 4 + ( f 4 ) ) )
21 = ( ( ( f 4 ) + ( 4 / 4 ) ) - 4 )
22 = ( ( ( f 4 ) + ( f 4 ) ) - ( ( f 4 ) + ( w 4 ) ) )
23 = ( ( ( f 4 ) + ( 4 / 4 ) ) - ( w 4 ) )
24 = ( ( 4 * 4 ) + ( 4 + 4 ) )
25 = ( ( ( f 4 ) + ( w 4 ) ) - ( 4 / 4 ) )
26 = ( ( ( f 4 ) + ( f 4 ) ) - ( ( f 4 ) - ( w 4 ) ) )
27 = ( ( 4 + ( f 4 ) ) - ( 4 / 4 ) )
28 = ( ( ( f 4 ) + ( f 4 ) ) - ( ( f 4 ) - 4 ) )
29 = ( ( 4 / 4 ) + ( 4 + ( f 4 ) ) )
30 = ( ( ( f 4 ) + ( f 4 ) ) - ( ( q 4 ) + ( w 4 ) ) )
31 = ( ( ( q 4 ) + ( q 4 ) ) - ( 4 / 4 ) )
32 = ( ( 4 * 4 ) + ( 4 * 4 ) )
33 = ( ( ( q 4 ) + ( q 4 ) ) + ( 4 / 4 ) )
34 = ( ( ( f 4 ) + ( f 4 ) ) - ( ( q 4 ) - ( w 4 ) ) )
35 = ( ( q ( ( f 4 ) / 4 ) ) - ( 4 / 4 ) )
36 = ( ( 4 * ( q 4 ) ) - ( 4 + ( f 4 ) ) )
37 = ( ( q ( ( f 4 ) / 4 ) ) + ( 4 / 4 ) )
38 = ( ( 4 * ( q 4 ) ) - ( ( f 4 ) + ( w 4 ) ) )
39 = ( ( ( q 4 ) + ( f 4 ) ) - ( 4 / 4 ) )
40 = ( ( 4 * ( 4 * 4 ) ) - ( f 4 ) )
41 = ( ( ( q 4 ) + ( f 4 ) ) + ( 4 / 4 ) )
42 = ( ( 4 * ( q 4 ) ) - ( ( f 4 ) - ( w 4 ) ) )
43 = ( ( ( q ( ( q 4 ) - ( w 4 ) ) ) - ( f 4 ) ) / 4 )
44 = ( ( 4 * ( q 4 ) ) - ( ( f 4 ) - 4 ) )
45 = ( ( ( q ( ( q 4 ) - ( w 4 ) ) ) - ( q 4 ) ) / 4 )
46 = ( ( 4 * ( q 4 ) ) - ( ( q 4 ) + ( w 4 ) ) )
47 = ( ( ( f 4 ) + ( f 4 ) ) - ( 4 / 4 ) )
48 = ( ( ( f 4 ) * ( 4 + 4 ) ) / 4 )
49 = ( ( ( f 4 ) + ( f 4 ) ) + ( 4 / 4 ) )
50 = ( ( 4 * ( q 4 ) ) - ( ( q 4 ) - ( w 4 ) ) )
```

6.2 Lösung zu Aufgabe (2)

Mit den Operatoren +, -, *, /, und sqrt() lassen sich nur die Zahlen von 1 bis 18 lückenlos berechnen. Danach entstehen schnell große Lücken. Die letzte berechenbare Zahl ist 1936.

```
 1 = ( 44 / 44 )
 2 = ( ( 4 * 4 ) / ( 4 + 4 ) )
 3 = ( ( 4 + ( 4 / 4 ) ) - ( w 4 ) )
 4 = ( ( ( w 4 ) * ( 4 + 4 ) ) / 4 )
 5 = ( ( 4 + ( w 4 ) ) - ( 4 / 4 ) )
 6 = ( w ( 44 - ( 4 + 4 ) ) )
```



```
  7 = ( ( 4 + 4 ) - ( 4 / 4 ) )
  8 = ( ( 4 * 4 ) - ( 4 + 4 ) )
  9 = ( ( 44 / 4 ) - ( w 4 ) )
 10 = ( ( 44 - 4 ) / 4 )
 11 = ( ( 44 / ( w 4 ) ) / ( w 4 ) )
 12 = ( ( 4 + 44 ) / 4 )
 13 = ( ( w 4 ) + ( 44 / 4 ) )
 14 = ( ( ( w 4 ) + ( 4 * 4 ) ) - 4 )
 15 = ( 4 + ( 44 / 4 ) )
 16 = ( ( 4 * 4 ) / ( 4 / 4 ) )
 17 = ( ( 4 / 4 ) + ( 4 * 4 ) )
 18 = ( ( 44 / ( w 4 ) ) - 4 )
 20 = ( ( 44 - 4 ) / ( w 4 ) )
 21 = ( ( 44 - ( w 4 ) ) / ( w 4 ) )
 22 = ( ( w 4 ) * ( 44 / 4 ) )
 23 = ( ( 44 + ( w 4 ) ) / ( w 4 ) )
 24 = ( ( 4 + 44 ) / ( w 4 ) )
 26 = ( 4 + ( 44 / ( w 4 ) ) )
 28 = ( 44 - ( 4 * 4 ) )
 30 = ( ( ( w 4 ) * ( 4 * 4 ) ) - ( w 4 ) )
 32 = ( ( 4 * 4 ) + ( 4 * 4 ) )
 34 = ( ( ( w 4 ) * ( 4 * 4 ) ) + ( w 4 ) )
 36 = ( 44 - ( 4 + 4 ) )
 38 = ( ( 44 - 4 ) - ( w 4 ) )
 40 = ( ( 44 - ( w 4 ) ) - ( w 4 ) )
 42 = ( ( w 4 ) + ( 44 - 4 ) )
 43 = ( 44 - ( 4 / 4 ) )
 44 = ( ( 4 * 44 ) / 4 )
 45 = ( 44 + ( 4 / 4 ) )
 46 = ( ( 4 + 44 ) - ( w 4 ) )
 48 = ( ( w 4 ) + ( 44 + ( w 4 ) ) )
 50 = ( ( w 4 ) + ( 4 + 44 ) )
 52 = ( 4 + ( 4 + 44 ) )
 56 = ( 4 * ( ( 4 * 4 ) - ( w 4 ) ) )
 60 = ( 44 + ( 4 * 4 ) )
 62 = ( ( 4 * ( 4 * 4 ) ) - ( w 4 ) )
 64 = ( ( 4 + 4 ) * ( 4 + 4 ) )
 66 = ( ( w 4 ) + ( 4 * ( 4 * 4 ) ) )
 68 = ( 4 + ( 4 * ( 4 * 4 ) ) )
 72 = ( 4 * ( ( w 4 ) + ( 4 * 4 ) ) )
 80 = ( ( w 4 ) * ( 44 - 4 ) )
 84 = ( ( w 4 ) * ( 44 - ( w 4 ) ) )
 86 = ( ( 44 * ( w 4 ) ) - ( w 4 ) )
 88 = ( 44 + 44 )
 90 = ( ( 44 * ( w 4 ) ) + ( w 4 ) )
 92 = ( ( w 4 ) * ( 44 + ( w 4 ) ) )
 96 = ( ( w 4 ) * ( 4 + 44 ) )
128 = ( ( 4 * 4 ) * ( 4 + 4 ) )
160 = ( 4 * ( 44 - 4 ) )
168 = ( 4 * ( 44 - ( w 4 ) ) )
172 = ( ( 4 * 44 ) - 4 )
174 = ( ( 4 * 44 ) - ( w 4 ) )
176 = ( ( 44 * ( w 4 ) ) * ( w 4 ) )
178 = ( ( 4 * 44 ) + ( w 4 ) )
180 = ( 4 + ( 4 * 44 ) )
184 = ( 4 * ( 44 + ( w 4 ) ) )
192 = ( 4 * ( 4 + 44 ) )
256 = ( ( 4 * 4 ) * ( 4 * 4 ) )
264 = ( 44 * ( 4 + ( w 4 ) ) )
352 = ( ( 4 * 44 ) * ( w 4 ) )
704 = ( 4 * ( 4 * 44 ) )
1936 = ( 44 * 44 ) )
```

6.3 Lösung zu Aufgabe (3)

Die gravierendste Einschränkung ergibt sich durch Weglassen der Division. Dadurch können keine ungeraden Zahlen berechnet werden. Am besten schneidet das Weglassen der Fakultät oder der Addition ab. Aber in keinem Fall gelang es, durch eine andere Kombination von Operatoren, die Zahlenreihe bis 50 vollständig zu berechnen.

```
 1 = ( ( 4 * 4 ) / ( 4 * 4 ) )
 2 = ( ( 4 * 4 ) / ( 4 + 4 ) )
 3 = ( ( ( w 4 ) + ( q 4 ) ) / ( 4 + ( w 4 ) ) )
 4 = ( ( ( q 4 ) * ( q 4 ) ) / ( 4 * ( q 4 ) ) )
 5 = ( ( 4 + ( 4 * 4 ) ) / 4 )
 6 = ( ( ( 4 + 4 ) + ( q 4 ) ) / 4 )
 7 = ( ( 4 + 4 ) - ( 4 / 4 ) )
 8 = ( ( 4 * 4 ) - ( 4 + 4 ) )
 9 = ( ( 4 / 4 ) + ( 4 + 4 ) )
10 = ( ( 4 + ( 4 * 4 ) ) / ( w 4 ) )
11 = ( ( q 4 ) - ( 4 + ( 4 / 4 ) ) )
12 = ( ( ( q 4 ) + ( q 4 ) ) - ( 4 + ( q 4 ) ) )
13 = ( ( ( 4 / 4 ) + ( q 4 ) ) - 4 )
14 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( w 4 ) + ( q 4 ) ) )
15 = ( ( 4 * 4 ) - ( 4 / 4 ) )
16 = ( ( 4 * 4 ) / ( 4 / 4 ) )
17 = ( ( 4 / 4 ) + ( 4 * 4 ) )
18 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( q 4 ) - ( w 4 ) ) )
19 = ( ( 4 + ( q 4 ) ) - ( 4 / 4 ) )
20 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( q 4 ) - 4 ) )
21 = ( ( 4 + ( q 4 ) ) + ( 4 / 4 ) )
22 = ( ( ( 4 + 4 ) + ( q 4 ) ) - ( w 4 ) )
23 = ( ( q ( 4 + ( 4 / 4 ) ) ) - ( w 4 ) )
24 = ( ( 4 * 4 ) + ( 4 + 4 ) )
25 = ( ( q ( 4 + ( q 4 ) ) ) / ( 4 * 4 ) )
26 = ( ( ( q 4 ) + ( q 4 ) ) - ( 4 + ( w 4 ) ) )
27 = ( ( q ( ( w 4 ) + ( q 4 ) ) ) / ( ( q 4 ) - 4 ) )
28 = ( ( ( 4 * 4 ) + ( q 4 ) ) - 4 )
29 = ( 4 + ( q ( 4 + ( 4 / 4 ) ) ) )
30 = ( ( ( 4 * 4 ) + ( q 4 ) ) - ( w 4 ) )
31 = ( ( ( q 4 ) + ( q 4 ) ) - ( 4 / 4 ) )
32 = ( ( 4 * 4 ) + ( 4 * 4 ) )
33 = ( ( 4 / 4 ) + ( ( q 4 ) + ( q 4 ) ) )
34 = ( ( w 4 ) + ( ( 4 * 4 ) + ( q 4 ) ) )
35 = ( ( q ( 4 + ( w 4 ) ) ) - ( 4 / 4 ) )
36 = ( 4 + ( ( 4 * 4 ) + ( q 4 ) ) )
37 = ( ( q ( 4 + ( w 4 ) ) ) + ( 4 / 4 ) )
38 = ( ( 4 + ( w 4 ) ) + ( ( q 4 ) + ( q 4 ) ) )
39 = --------------------------------------------------
40 = ( ( 4 + 4 ) + ( ( q 4 ) + ( q 4 ) ) )
41 = ( ( q 4 ) + ( q ( 4 + ( 4 / 4 ) ) ) )
42 = ( ( ( q ( q 4 ) ) - 4 ) / ( 4 + ( w 4 ) ) )
43 = ( ( ( q ( q 4 ) ) + ( w 4 ) ) / ( 4 + ( w 4 ) ) )
44 = ( ( 4 * ( q 4 ) ) - ( 4 + ( q 4 ) ) )
```


45 = (((q ((q 4) - (w 4))) - (q 4)) / 4)
46 = ((4 * (q 4)) - ((w 4) + (q 4)))
47 = ((((q (q 4)) - 4) / 4) - (q 4))
48 = ((4 * (4 * 4)) - (q 4))
49 = (q ((4 + 4) - (4 / 4)))
50 = ((4 * (q 4)) - ((q 4) - (w 4)))

6.4 Lösung zu Aufgabe (6)

Hier meine Frage: Welche Zahl ist die erste nicht berechenbare Zahl in aufsteigender Reihenfolge?

Antwort: Nach meinen Berechnungen ist das die Zahl 91.

6.5 Lösung zu Aufgabe (5)

Auch negative Zahlen lassen sich lückenlos bis -50 berechnen.

-50 = (((q 4) - (w 4)) - (4 * (q 4)))
-49 = ((q (f 4)) - (q ((4 / 4) + (f 4))))
-48 = (((f 4) * (f 4)) / (4 - (q 4)))
-47 = ((4 / 4) - ((f 4) + (f 4)))
-46 = (((w 4) + (q 4)) - (4 * (q 4)))
-45 = ((f ((f 4) / 4)) / ((f (q (f 4))) - (q4)))
-44 = (((w 4) - 4) * ((f 4) - (w 4)))
-43 = ((w 4) - ((f ((f 4) / 4)) / (q 4)))
-42 = (((f 4) - (w 4)) - (4 * (q 4)))
-41 = (((q (f 4)) - (w 4)) / ((w 4) - (q4)))
-40 = ((f 4) - (4 * (4 * 4)))
-39 = ((4 / 4) - ((q 4) + (f 4)))
-38 = (((w 4) + (f 4)) - (4 * (q 4)))
-37 = (((q (f 4)) + (q 4)) / ((f (q (f 4))) - (q 4)))
-36 = ((q (f 4)) / ((4 + 4) - (f 4)))
-35 = (((q (q 4)) + (f 4)) / ((q 4) - (f 4)))
-34 = (((q 4) - (w 4)) - ((f 4) + (f 4)))
-33 = ((q (q 4)) - (q ((4 / 4) + (q 4))))
-32 = (((q 4) * (f 4)) / (4 - (q 4)))
-31 = ((4 / 4) - ((q 4) + (q 4)))
-30 = (((w 4) + (q 4)) - ((f 4) + (f 4)))
-29 = ((4 + (q (f 4))) / (4 - (f 4)))
-28 = (((f 4) - 4) - ((f 4) + (f 4)))
-27 = ((4 / 4) - (4 + (f 4)))
-26 = (((f 4) - (w 4)) - ((f 4) + (f 4)))
-25 = ((4 / 4) - ((w 4) + (f 4)))
-24 = ((q (f 4)) / ((4 - 4) - (f 4)))
-23 = ((w 4) - ((4 / 4) + (f 4)))
-22 = (((w 4) + (f 4)) - ((f 4) + (f 4)))
-21 = (4 - ((4 / 4) + (f 4)))
-20 = (((q 4) + (f 4)) / ((w 4) - 4))
-19 = ((4 + (4 / 4)) - (f 4))
-18 = (((f 4) - (w 4)) - ((q 4) + (f 4)))
-17 = ((4 / 4) - ((w 4) + (q 4)))
-16 = ((4 - 4) - (4 * 4))
-15 = ((4 / 4) - (4 * 4))
-14 = ((4 + (f 4)) / ((w 4) - 4))
-13 = (((w 4) + (f 4)) / ((w 4) - 4))
-12 = ((4 * (f 4)) / ((q 4) - (f 4)))
-11 = (((f 4) - (w 4)) / ((w 4) - 4))
-10 = (((f 4) - (w 4)) - ((q 4) + (q 4)))
-9 = (((w 4) + (q 4)) / ((w 4) - 4))
-8 = ((4 + 4) - (4 * 4))
-7 = ((4 / 4) - (4 + 4))
-6 = (((f 4) + (f 4)) / ((q 4) - (f 4)))
-5 = (((q 4) + (f 4)) / ((q 4) - (f 4)))
-4 = (((f 4) + (f 4)) / (4 - (q 4)))
-3 = ((f 4) / ((4 * 4) - (f 4)))
-2 = (((q 4) + (f 4)) / (4 - (f 4)))
-1 = ((4 - 4) - (4 / 4))
0 = ((4 * 4) - (4 * 4))

6.6 Lösung zu Aufgabe (7)

Insgesamt ließen sich für die Zahlen von 1 bis 50 etwas mehr als 30000 Varianten berechnen. Nachfolgend einige Beispiele: alle Varianten zu den Zahlen 39, 33 und 31.

39 = (((q 4) + (f 4)) - ((w 4) / (w 4)))
39 = (((q 4) + (f 4)) - ((q 4) / (q 4)))
39 = (((q 4) + (f 4)) - ((f 4) / (f 4)))
39 = (((q 4) + (f 4)) - (4 / 4))
39 = (((q 4) - (4 / 4)) + (f 4))
39 = ((q 4) + ((f 4) - (4 / 4)))
39 = ((f 4) + ((q 4) - ((f 4) / (f 4))))
39 = ((f 4) + ((q 4) - ((q 4) / (q 4))))
39 = ((f 4) + ((q 4) - ((w 4) / (w 4))))
39 = ((q 4) + ((f 4) - ((w 4) / (w 4))))
39 = ((q 4) + ((f 4) - ((q 4) / (q 4))))
39 = ((q 4) + ((f 4) - ((f 4) / (f 4))))

33 = (((w 4) / (w 4)) + ((q 4) + (q 4)))
33 = (((q 4) / (q 4)) + ((q 4) + (q 4)))
33 = (((f 4) / (f 4)) + ((q 4) + (q 4)))
33 = ((4 / 4) + ((q 4) + (q 4)))
33 = (((w 4) * (q 4)) + ((w 4) / (w 4)))
33 = (((w 4) * (q 4)) + ((q 4) / (q 4)))
33 = (((w 4) * (q 4)) + ((f 4) / (f 4)))
33 = (((w 4) * (q 4)) + (4 / 4))
33 = (((4 / 4) + (q 4)) + (q 4))
33 = ((q ((f 4) / (4 + 4))) + (f 4))
33 = ((q (4 - (4 / 4))) + (f 4))
33 = ((q ((4 / 4) + (w 4))) + (f 4))
33 = ((f 4) + (((w 4) + (q 4)) / (w 4)))
33 = ((q 4) + (((w 4) / (w 4)) + (q 4)))
33 = ((q 4) + ((q 4) + ((q 4) / (q 4))))
33 = ((q 4) + ((q 4) + ((f 4) / (f 4))))
33 = ((q (((q 4) - (w 4)) / (w 4))) - (q 4))
33 = ((q ((4 + (f 4)) / 4)) - (q 4))
33 = ((f 4) + (q (((f 4) + (f 4)) / (q 4))))
33 = ((f 4) + (q (((w 4) * (f 4)) / (q 4))))
33 = ((f 4) + (q ((f 4) / (4 * (w 4)))))
33 = ((f 4) + (q ((f 4) / ((q 4) / (w 4)))))
33 = ((f 4) + (q ((f 4) / ((f 4) - (q 4)))))
33 = ((f 4) + (q (((f 4) / (w 4)) / 4)))
33 = ((f 4) + (q (((q 4) - 4) / 4)))
33 = ((f 4) + (q (((f 4) / 4) / (w 4))))
33 = ((f 4) + (q ((4 + (w 4)) / (w 4))))

```
33 = ( ( f 4 ) + ( q ( 4 - ( ( w 4 ) / ( w 4 ) ) ) ) )
33 = ( ( f 4 ) + ( q ( 4 - ( ( q 4 ) / ( q 4 ) ) ) ) )
33 = ( ( f 4 ) + ( q ( 4 - ( ( f 4 ) / ( f 4 ) ) ) ) )
33 = ( ( f 4 ) + ( q ( ( ( w 4 ) / ( w 4 ) ) + ( w 4 ) ) ) )
33 = ( ( f 4 ) + ( q ( ( w 4 ) + ( ( q 4 ) / ( q 4 ) ) ) ) )
33 = ( ( f 4 ) + ( q ( ( w 4 ) + ( ( f 4 ) / ( f 4 ) ) ) ) )
33 = ( ( f 4 ) + ( ( q ( ( f 4 ) / 4 ) ) / 4 ) )
33 = ( ( f 4 ) + ( ( q ( 4 + ( w 4 ) ) ) / 4 ) )

31 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( w 4 ) / ( w 4 ) ) )
31 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( q 4 ) / ( q 4 ) ) )
31 = ( ( ( q 4 ) + ( q 4 ) ) - ( ( f 4 ) / ( f 4 ) ) )
31 = ( ( ( q 4 ) + ( q 4 ) ) - ( 4 / 4 ) )
31 = ( ( ( w 4 ) * ( q 4 ) ) - ( ( w 4 ) / ( w 4 ) ) )
31 = ( ( ( w 4 ) * ( q 4 ) ) - ( ( q 4 ) / ( q 4 ) ) )
31 = ( ( ( w 4 ) * ( q 4 ) ) - ( ( f 4 ) / ( f 4 ) ) )
31 = ( ( ( w 4 ) * ( q 4 ) ) - ( 4 / 4 ) )
31 = ( ( ( q 4 ) - ( 4 / 4 ) ) + ( q 4 ) )
31 = ( ( f 4 ) + ( ( ( q 4 ) - ( w 4 ) ) / ( w 4 ) ) )
31 = ( ( f 4 ) + ( ( 4 + ( f 4 ) ) / 4 ) )
31 = ( ( q 4 ) + ( ( q 4 ) - ( ( f 4 ) / ( f 4 ) ) ) )
31 = ( ( q 4 ) + ( ( q 4 ) - ( ( q 4 ) / ( q 4 ) ) ) )
31 = ( ( q 4 ) + ( ( q 4 ) - ( ( w 4 ) / ( w 4 ) ) ) )
```

RINGTAUSCH ausgeschlossen...

Lösung einer alten Aufgabe

Friederich Prinz

Friederich.Prinz@t-online.de

1994 hat Arndt Klingelnberg den sporadischen Reigen der Knobelaufgaben von Forthern für Forther mit einer Aufgabe eröffnet, die er für etwas „ausgefuchstere“ Forther formuliert hatte. Die Inhalte zweier Variablen sollten miteinander vertauscht werden, ohne den üblichen Ringtausch, ohne Zuhilfenahme des Stacks oder des Returnstacks und ohne Zuhilfenahme sonstiger Speicherplätze. Auch 'Ringtauschoperationen' innerhalb der CPU-Register sollen nicht zur Anwendung kommen, ebensowenig wie der XCHG-Befehl der 80x86 Prozessoren.

Ich hatte mir damals, als die Forthgesellschaft gerade 10 Jahre jung war, eine Lösung einfallen lassen, die aber, vermutlich gleichzeitig mit anderen Lösungen, irgendwie verloren gegangen ist und darum nie veröffentlicht wurde. Vor wenigen Wochen bin ich aber nach der Lösung für genau diese Aufgabe gefragt worden. Nun – hier ist mein Lösungsvorschlag aus 1994.

Wenn man an den einfachen (und einleuchtenden) Ringtausch erst einmal gewöhnt ist, dann fällt es vermutlich schwer, andere Lösungen zu finden. Das XCHG-Verbot erschwert die Sache zusätzlich, zumindest wenn man ´aus der PC-Welt kommt´. Haben andere als 80x86 CPUs entsprechende Befehle? Ich weiß das nicht...

Die Moerser ´Credos´ "Es liegt alles im Speicher" und "Alles was im Speicher liegt, sind Bitmuster" bringen den Tüftler dennoch schnell auf den richtigen Weg. Schließlich sagt man uns FORTHern nicht umsonst nach, daß wir ´manische Bitknipser´ seien. Also - erst einmal kurz nachgedacht:

Wenn man ein Bitmuster nach links verschiebt, dann fällt das am weitesten links stehende Bit heraus. Das ´Schieben nach links´ entspricht einer Multiplikation mit 2. Für die Multiplikation stellt ZF das Wort 2* zur Verfügung (2 * tut es natürlich genauso, wenn das eigene System 2* nicht kennt). Wenn man zwei Variablen definiert, meinetwegen VARIABLE Eins und VARIABLE Zwei, dann läßt sich folgendes in Gedanken nachvollziehen: Der Inhalt von VARIABLE Eins wird nach links geschoben, ebenso der Inhalt von VARIABLE Zwei. Das links bei Eins herausgefallene Bit wird rechts bei Zwei einfach wieder eingeschoben. Und das links bei Zwei herausgefallene Bit wird bei Eins wieder eingeschoben. Wenn ich das Ganze 16 Mal mache, weil ich von 16-Bit Inhalten ausgehe, dann sollten die jeweils rechts eingeschobenen Bits am Ende ihre ursprünglichen Werte repräsentieren, allerdings unter dem jeweils anderen Speicherplatz - sprich: unter dem jeweils anderen Namen.

Wie weiß ich aber, ob bei einer Schiebeoperation tatsächlich ein Bit aus dem Muster herauskippt ? Wollte ich die Knobelei mit ZF´s Assembler realisieren, könnte ich das herausfallende Bit sehr einfach aus dem CARRY-Bit des Flagregisters entnehmen, bzw. gleich die gesamte Schiebeoperation ´durch das Carry´ organisieren. Aber ich will es in HighLevel Forth versuchen. Vielleicht wollen FORTHer mit anderen Systemen die Lösung ja nachvollziehen... Nun, herausfallen kann nur das, was ´am Rand steht´. Ich muß eigentlich nur prüfen, ob das jeweils ganz links stehende Bit gesetzt ist oder nicht. Ist dies der Fall, muß in der ´Gegenvariablen´ rechts ein Bit eingefügt werden, also nach der Schiebeoperation (!) eine ´1´ addiert werden. Ich muß vor der Schiebeoperation prüfen, was ich nachher einfügen muß !

Das höchstwertigste Bit, also das Bit ´ganz links´ entspricht dem dezimalen Wert 32768. Wenn ich ein beliebiges Muster aus 16 Bits mit 32768 AND verknüpfe, dann erhalte ich 32768 als Ergebnis, wenn eben dieses Bit gesetzt ist...

Schauen wir einfach einmal nach, ob das so funktioniert...

```
VARIABLE Eins  4  Eins !
VARIABLE Zwei  17 Zwei !
```

Naja, mir sind „damals“ eben keine passenderen Variablennamen eingefallen...

```
: Tauschen ( -- )
    16 0 DO      \ 16 Mal muß die Schleife
                 \ arbeiten, für 16 Bits
    Eins @       \ Inhalt von 'Eins' lesen
    32768 AND    \ und auf das höchstwertigste
                 \ Bit testen
    32768 = IF 1 \ Wenn das Bit gesetzt ist,
                 \ dann eine '1' auf
    ELSE 0       \ dem Stack festhalten,
    THEN         \ sonst '0' merken
    Zwei @       \ Was für 'Eins' gilt, muß
                 \ auch bei 'Zwei'
    32768 AND    \ durchgeführt werden.
    32768 = IF 1
    ELSE 0
    THEN
    Eins @ 2*    \ 'Eins' nochmal auslesen und
                 \ 'links schieben',
    +            \ das von 'Zwei'gekommene Bit
                 \ rechts 'einfügen'
    Eins !       \ und das neue Bitmuster nach
                 \ 'Eins' speichern.
    Zwei @ 2*    \ ...natürlich gilt für
                 \ 'Zwei' wieder das Gleiche,
    +            \ mit dem Unterschied, daß
                 \ hier das von 'Eins'
    Zwei !       \ gekommene Bit eingefügt
                 \ wird.
    LOOP ;       \ Woala - wie der Franzose zu
                 \ sagen pflegt ;-)
```

Selbstverständlich funktioniert das so (war ja auch ordentlich durchdacht - und ließ sich mit ZFs DEBUG Schritt für Schritt verfolgen und ausprobieren :-]) !

Und selbstverständlich läßt sich das Ganze auch wesentlich ´kompakter´ definieren. Aber so ist es sicher verständlicher - auch für Hobbyisten wie mich selbst.

Und wie haben Sie diese Aufgabe „damals“ gelöst?

Betreff: **Chemnitzer Linuxtag 2004**
Von: Enrico Haertel <enrico.haertel@s2000.tu-chemnitz.de>

Sehr geehrte Damen und Herren,

ich schreibe Ihnen im Rahmen der Vorbereitungen zum Chemnitzer Linuxtag am **06./07. März 2004.**

http://www.tu-chemnitz.de/linux/tag

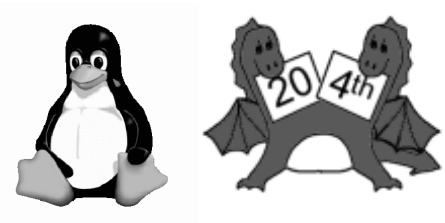
Auf Hinweis von Sven Guckes habe ich mir die Forth-Website angesehen. Ich schreibe Ihnen nun, weil wir der Meinung sind, daß der Forth e.V. auf den Chemnitzer Linuxtag mindestens genauso gut passt, wie auf den in Karlsruhe.

Im kommenden Jahr möchten wir verstärkt die Interessen von Ingenieuren und Naturwissenschaftlern ansprechen und suchen noch nach interessanten Themen. Die Programmiersprache Forth und insbesondere ihre Anwendung würde uns daher sehr gut ins Bild passen.

Wir möchten Sie also einladen, auch beim Chemnitzer Linuxtag mit einem kleinen Stand dabei zu sein.
Für eventuelle Rückfragen und natürlich auch für genauere Absprachen stehe ich Ihnen gerne per mail zur Verfügung.

Wir würden uns freuen, von Ihnen zu hören.

Mit freundlichen Grüßen
Enrico Härtel



Sind das nun zwei oder drei Kumpel? Jedenfalls passen Swap und der Linux-Pinguin nicht schlecht zusammen. Und wenn Sie meine Frage mit einem kompetenten Fachmann erörtern möchten, dann fahren Sie hin, zum Chemnitzer Linux-Tag. Sie treffen dort zumindest Carsten Strotmann und vermutlich noch weitere Mitglieder der Forthgesellschaft.

fep

Hallo, Allerseits!

Falls Ihr Euch wundert: Ich habe nichts über die SVFIG Treffen vom Oktober und November geschrieben, da ich bis nach Donald Knuth's Vorlesungen im "Computer History Museum" warten wollte, um alle meine Berichte in einem Paket zusammenzufassen und um mit den besten Grüßen an alle für das sich schnell nähernde Jahr 2004 zu schließen.

Am 23. Oktober verbrachte unsere übliche Kern-Gruppe von ungefähr 15 (plus/minus drei) Mitgliedern den Vormittag damit, Dr. Ting's Beschreibung eines neuen "aufgebohrten" 8051 Boards, welches in Taiwan entwickelt wurde, zu hören und wie eP32, eForth, F#, ein RGB-Display, FML, etc. dieses Board möglicherweise zum "zukünftigen chinesischen Computer" machen könnte. Am Nachmittag sahen wir Dwight Elvey's Folienpräsentation über die kürzliche Vintage Computer Messe und eine Reihe Tips darüber, wie man wirklich alte Computer booten und alte Programme retten und wiederherstellen kann.

Der jährliche "Forth-Tag" kam am 22. November, und ich muß gestehen, daß es eine angenehme Überraschung war, an die 30 Leute zu Beginn zu zählen. Kevin Apert hatte eine wunderbare Arbeit geleistet, um viele Forther, die man eine lange Zeit nicht gesehen hat, aufzuspüren, anzusprechen und einzuladen. Die Enttäuschung bestand darin, daß es wohl das erste Jahr war, an dem Chuck Moore nicht anwesend war, um uns mit seinem

"Geplauder am Feuer" zu unterhalten. Er und seine Frau sind vor etwas über einem Jahr nach Sierra City (auf 4500 Fuss Höhe) in die kalifornischen Sierras umgezogen. Jeff Fox, der den Berg hinaufgestiegen war, berichtete von seinen Problemen, mit den Moores Schritt zu halten, während sie alle durch den Schnee wanderten.

Ich bin sicher, daß die SVFIG Webseite den Zeitplan und die Themen der Vorträge der acht Redner zeigt. Alle schienen mehr zu anzubieten zu haben, als es die Zeit erlaubte. Daher werde ich mich auf ein paar Dinge von nicht technischem Inhalt beschränken.

Der etwas kalte Wind konnte nicht verhindern, daß Ting's übliches Grillen im Freien ein erfolgreiches geselliges Ereignis wurde. Die neuen Gesichter unter den Vortragenden waren Randy Thelen mit seinem selbstgebauten TTL-Board, auf dem Mipy ("million instructions per year") lief, und Joseph M. O'Connor, der quer durch das Land aus Connecticut gekommen war, um Creole, eine Forth-artige Skriptsprache in Borland Delphi, vorzustellen. La Farr Stuart kam wieder, um einige seiner Freunde aus den 1970ern zu treffen und uns mit Forth in der Mathematik zu unterhalten. Jay Melvin nahm für sich in Anspruch, der erste Kunde von Tom Zimmer gewesen zu sein und den Kauf ZForth genannt zu haben. Tom war einer der alten Forther, die nicht kommen konnten, aber er hatte auf Kevin's Einladung mit einer Email geantwortet. Die einzige Beschwerde dieses Tages bestand darin, daß die Fonts der PowerPoint Präsentationen unserer Gruppe nicht entgegenkamen, die mit den Jahren zunehmend kurzsichtig wird.

Das Museum für Computer-Geschichte scheint sich an seinem neuen Platz in Mountain View, Kalifonien, gut zu entwickeln. Auch wenn ich mich wiederhole, es lohnt sich

www.computerhistory.org

zu besuchen, um mehr über die Ausstellungen und verschiedenen Aktivitäten zu erfahren.

Im Buch von Sasha und Lazere "Out of their Minds - The Lives and Discoveries of 15 Great Computer Scientists" erscheint das folgende Zitat von Donald Knuth gleich zweimal: "It's not true that necessity is the only mother or father of invention. The other part is that a person has to have the right background for the problem. I don't just go around working on every problem that I see. The ones I solve, I say, 'Oh, man, I have a unique background that might let me solve it--it's my destiny, my responsibility.'" ("Es ist nicht war, daß Notwendigkeit die einzige Mutter oder der einzige Vater einer Erfindung sind. Der andere Teil besteht darin, daß eine Person den richtigen Hintergrund für das Problem haben muß. Ich laufe nicht einfach herum und arbeite an jedem Problem, welches ich sehe. Für die, die ich löse, sage ich mir: 'Oh Mann, ich habe einen einzigartigen Hintergrund der mich in die Lage versetzt, dieses Problem zu lösen -- Es ist mein Schicksal, meine Pflicht.'"

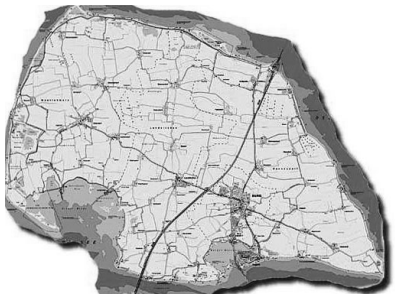
Das Kapitel über Knuth endet mit der Aussage, daß der "Volksmund glaubt, daß Knuth der größte Computerprogrammierer aller Zeiten ist." Das Buch wurde 1995 geschrieben und so klingt inzwischen das Wort "Programmierer" nicht ruhmreich genug für die jungen Grünschnäbel, die gerade einmal durchs College kommen und Graduierungen in Computerwissenschaften verliehen bekommen.

Ich sehe Knuth als ein Genie in Mathematik, Musik und Computerwissenschaften, die alle, wie er sagt, die Wissenschaft der Muster gemeinsam haben. Wenn ich mir seine Bücher anschaue, erkenne ich, daß er recht hat -- daß nur ungefähr 2 Prozent der Bevölkerung mentale Qualitäten haben, die zu Computern passen, was sie schließlich als Computerwissenschaftler qualifiziert.

Knuth war letzten Mittwoch im Museum, um einen Vortrag über die Geschichte der Computersprachen zu halten und sein neues Buch "Ausgewählte Kapitel über Computersprachen" zu signieren. Insbesondere beschrieb er ungefähr ein Dutzend Vorläufer von Fortran, indem er zeigte, wie ein einfacher Algorithmus in jeder dieser verschiedenen Sprachen kodiert werden würde.
Der Punkt, den er hervorhob, war, daß wir bei den Grundkonzepten von Programmiersprachen, wie wir sie heute kennen, nicht angekommen wären, ohne eine Menge harter Arbeit brillianter und gebildeter Leute.
Die Namensliste von Zuse bis Backus und die Namen der verschiedenen Sprachen, über die er sprach, würden zuviel Platz in diesem Brief benötigen. Ich habe verstanden, daß all dies in seinem neuen Buch gut behandelt wird. Ich möchte nur noch eine abschließende Bemerkung darüber anfügen, daß Knuth (und sein früherer Student und Mitarbeiter an der TPK Algorithmendarstellung) farbkodierte Syntax in den Folien benutzten, um die Verständlichkeit der Programmbeispiele zu erhöhen. Das führt mich zu der Frage, ob ColorForth eines Tages wohl in der Geschichte der Computersprachen erwähnt werden wird. Happy New Year!

Henry

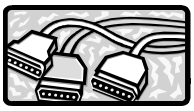
Übersetzung: *Thomas Beierlein*



Haben Sie sich schon für die Tagung auf Fehmarn angemeldet?

Wen Sie dort alles treffen, können Sie auf den WEB-Seiten der Forthgesellschaft nachlesen:

http://www.forth-ev.de/tagung/anmeldungen04.html



UART-Controller für HOLON-Forth

Ein

UART-Controller für HOLON-Forth

Friederich Prinz

Friederich.Prinz@t-online.de

Ich habe die Weihnachtsferien 2003 genutzt, um ein wenig mit LEGO zu spielen, genauer: um mit dem RCX zu spielen. Ich arbeite, sehr sporadisch, an einer Schnittstelle zum RCX für das HOLON-Forth. HOLON86 arbeitet zwar unter anderem mit einem Target an einer seriellen Schnittstelle, bringt aber zunächst keinen separaten „Com-Treiber“ mit. Auf Anfrage stellt Wolf Wejgaard gerne ein Modul mit entsprechenden Worten zur Verfügung. Diese Worte arbeiten leider nicht mit meinem System zusammen. Insbesondere die Initialisierung der Schnittstelle wird nicht korrekt durchgeführt. Ich vermute, daß WinXP hierfür verantwortlich ist. Ich habe aber nicht lange nach möglichen Ursachen gesucht, sondern lieber gleich Abhilfe geschaffen, und mir den benötigten „Treiber“ selbst geschrieben.
Das Modul steht bei mir zum Download bereit. Ich schicke es per mail gerne als *.mod oder *.mtx, wobei letztgenannte Variante als ASCII-Datei leicht für eventuelle Portierungen auf andere Forth-Systeme genutzt werden kann.
Nicht zuletzt um zu zeigen, wie „ordentlich“ HOLON-Module sozusagen zwangsweise organisiert sind, veröffentliche ich hier den Quelltext des Moduls so, wie er als *.mtx Modul zur Verfügung steht.

```
Module: UARTCtrl

Der UART-Controller ist ein einfacher Schnitt-
stellentreiber fuer COM-Ports in PCs.
Seine Quelltexte duerfen, wie unter Forth-
Programmierern ueblich, frei verwendet und
weitergegeben werden.
                        Friederich Prinz
                    Forthgesellschaft e.V.
                         [Ver. 2.1]


Group: Leutnant

Leutnant, der Platzhalter, der an der Stelle
des Befehlshaber steht, ermoeglicht es hier,
der ersten wirklichen Gruppenbezeichnung, die
dem Leutnant unmittelbar folgt, eine neue
Gruppe vorzusetzen.
Ohne den "Leutnant", der hier nur ein Dummy
ist, faellt es mir schwer, selbst im HOLON
Ordnung zu halten.

Group: Beschreibungen

Erklaerende Texte mit Hinweisen zur "richtigen
Bedienung" des UART-Controllers.

Word: Polling
Text: Polling

Der UART-Controller arbeitet im Polling, d.h.
ohne die BIOS-Funktionen des INT 14h. Die Re-
gister des UART werden direkt beschrieben und
gelesen. Das hat einen entscheidenden Grund:
In meinen Rechnern (PIII Katmai; WIN XP Prof.)
funktionieren die Init-Routinen 00 und 04 des
INT 14h nicht. Die Baudrate und die Angaben
zur Paritaet werden nicht korrekt in die UART-
Register geschrieben. Wenn ich das selbst ma-
chen muss, dann mache ich aber auch gleich al-
les selbst.

Word: Baudrate
Text: Baudrate

Sie koennen als Baudrate angeben, was immer
Sie wollen. 8736 BAUD werden ebenso akzeptiert
wie 2400 BAUD. Ob der Sender/Empfaenger auf
der anderen Seite mit dieser Baudrate etwas
anzufangen weiss, muessen Sie beurteilen. Ob
Sie das ueber eine Direktverbindung der Signa-
le BAUDOUT und RCLK (ReceiveClock) sicherstel-
len, oder ueber Einstellungen, die sie in ih-
ren Programmen vornehmen, bleibt Ihnen ueber-
lassen.

Word: COM-Parameter
Text: COM-Parameter aendern

Aenderungen an den Einstellungen der hier vor-
gegebenen COM-Parameter koennen Sie auf
(mindestens) drei Arten vornehmen:

1. Aendern und Rekompilieren des Wortes
   ComInit
2. Erstellen einer Kopie von ComInit mit
   geaenderten Parametern
3. Interaktiver Aufruf der Parameter, so wie
   in ComInit gezeigt.

Die Reihenfolge der Angaben ist dabei belang-
los!

Word: Vorteile
Text: Vorteile

Ein "pollender" Controller ist (hoffentlich)
weitgehend unabhaengig vom Rechner (solange
das ein PC ist) und vom Betriebssystem
(solange es Zugriffe auf die Hardware des UART
erlaubt). Ich gehe also davon aus, dass der
UART-Controller in allen PC-Systemen arbeitet.
Wenn es doch einmal nicht klappt, dann bitte
eine Info an
                    Friederich.Prinz@t-online.de

Word: Erweitern
Text: Erweitern

Ich habe den UART-Controller geschrieben, weil
ich eine kleine, leicht zu bedienende Schnitt-
stelle fuer HOLON-Forth (Wolf Weijgaard) zum RCX
(LEGO) haben wollte. Ich habe aber bereits beina-
he alles darin angelegt, was man benoetigt, um
die Schnittstelle zu erweitern. Zum Beispiel lie-
sse sich das Polling durch Tasks uebernehmen, die
sich ihre Steuersignale vom UART geben lassen.
Bitte schoen; einfach zugreifen – aber dann bitte
eine Kopie an mich schicken.
```

```
Word: Einstellungen
TEXT: Einstellungen
```

...braucht der UART-Controller eigentlich nicht. Wenn das Abholen der Daten zu langsam geschieht, oder wenn das Senden bei hohen Baudraten nicht schnell genug geschehen sollte, koennen die Parameter vor den TICKS? in den Worten _trn und _rcv entsprechend geaendert werden.

```
Word: Portierbarkeit-1
Text: Portierbarkeit-1
```

Der UART-Controller ist unter HOLON in ANS-Forth geschrieben. Seine Quellen sollten darum relativ leicht in andere FORTH-Systeme uebertragbar sein. Probleme koennten HOLONs Objekte bereiten, die in anderen FORTH-Systemen aber mit CREATE ... DOES> Konstrukten nachvollziehbar sind. Literatur hierzu findet sich fuer den deutschsprachigen Raum in diversen Veroeffentlichungen der "Vierte Dimension" (Organ der Forthgesellschaft e.V.).

```
Word: Portierbarkeit-2
Text: Portierbarkeit-2
```

Probleme bei der Uebertragung in ein anderes FORTH-System koennten auch durch die hier gewaehlte Reihenfolge der Worte entstehen. Unter HOLON ist es gleichgueltig, an welcher Stelle im Quelltext ein Wort definiert ist. Es kann auch waehrend der Kompilation von Worten aufgerufen werden, die zu einem frueheren Zeitpunkt, bzw. an einer frueheren Stelle im Quelltext definiert werden. Ich ordne die von mir definierten Worte im HOLON so, wie sie nach meiner Auffassung organisatorisch zusammen gehoeren.

```
Word: Pufferinhalte
Text: Pufferinhalte
```

Der UART-Controller faengt bewusst nicht viele Fehler ab.
Im Besonderen ist das aufrufende Programm selbst dafuer verantwortlich, dass nach jedem TRANSMIT und jedem RECEIVE die entsprechenden Puffer wieder geloescht, und die zugehoerigen Zeiger zurueckgesetzt werden.
Geschieht dies nicht, arbeiten Transmitter und Receiver bis an die jeweiligen Puffergrenzen und bleiben dort "stehen".

```
Word: Erklaerung
Text: Erklaerung
```

Selbstverstaendlich erfolgt jegliche Nutzung des UART-Controllers sozusagen auf eigene Gefahr. Ich weiss zwar wirklich nicht, was durch eine Fehlfunktion des Treibers kaputt gehen koennte (ausser Daten, vielleicht), aber warum soll ich fuer irgend etwas garantieren, was ich verschenke?

```
Group: Deklarationen
```

Datenworte, die der UART-Controller benoetigt.

Die Worte zu den INT-Registern sind der Vollstaendigkeit halber hier aufgenommen, werden aber beim Polling nicht genutzt. Sie koennen gegebenenfalls zur Definition INT-gesteuerter Treiber genutzt werden.

```
Word: _com1
$40 $00 @L INTEGER _com1

\ Basisadresse des COM-Ports; Wird waehrend
\ der Kompilation aus dem BIOS ausgelesen. Das
\ vereinfacht das langwierige Suchen nach DEM
\ Fehler. Da hat man alles richtig gemacht,
\ und bekommt die Schnittstelle doch nicht
\ initialisiert; Vielleicht, weil man in der
\ Adressangabe einen "Zahlendreher" hat, oder
\ weil der UART-Controller auf einem alten
\ PC-XT laufen soll. Da sind die Adressen
\ anders als beim AT...

Word: _com2
$40 $02 @L INTEGER _com2

Word: _com3
$40 $04 @L INTEGER _com3

Word: _com4
$40 $06 @L INTEGER _com4

Word: ComPort
_com2 INTEGER ComPort
\ Umschalten mit _comx IS ComPort

Word: IOBufReg
\ DLAB in LineControlRegister ist 0
: IOBufReg ( -- adr )
    ComPort ;

\ Sende- und Empfangsregister; Offset 00 auf
\ ComPort

Word: IntAkReg
\ DLAB in LineControlRegister ist 0
: IntAkReg ( -- adr )
    ComPort 1 + ;
\ Interrupt-Aktivierungsregister

Word: IntIDReg
: IntIDReg ( -- adr )
    ComPort 2 + ;

\ Interrupt-Identifizierungsregister

Word: LinCtReg
: LinCtReg ( -- adr )
    ComPort 3 + ;
\ Line-Controlregister

Word: ModCtReg
: ModCtReg ( -- adr )
    ComPort 4 + ;
\ Modem-Controlregister

Word: SerStReg
: SerStReg ( -- adr )
    ComPort 5 + ;
\ Serial-Statusregister
```


```
Word: ModStReg
: ModStReg ( -- adr )
    ComPort 6 + ;
\ Modem-Statusregister

Word: LSBLtReg
\   DLAB in LineControlRegister ist 1
: LSBLtReg ( -- adr )
    ComPort ;

\ LSBLatch(1)-Statusregister; DLAB in LinCtReg
\ muss gesetzt sein! Niederwertiges Register
\ zur Aufnahme der kodierten Baudrate.

Word: MSBLtReg
\   DLAB in LineControlRegister ist 1
: MSBLtReg ( -- adr )
    ComPort 1+ ;

\ MSBLatch(2)-Statusregister; DLAB in LinCtReg
\ muss gesetzt sein! Hoeherwertiges Register
\ zur Aufnahme der kodierten Baudrate.

Word: init?
0 INTEGER init?

\ Zugriffe auf nicht initialisiere ComPorts
\ fuehren zu derben Fehlern, weil Transmitter
\ und Receiver in Datenbereiche schreiben
\ wollen, die ohne Initialisierung nicht
\ vorhersehbar sind.
\ Der Transmitter darf nur arbeiten, wenn eine
\ Initialisierung diese Fehlerquelle beseitigt
\ hat.
```

Group: RS232Kosmetik

Ich teste mit einem RS232-Schnittstellentester, ob ich ueberhaupt Daten an den ComPort bringe, ob die Initialisierung des ComPorts klappt usw..
Wenn das erledigt ist, hat der Tester nur noch "kosmetische" Aufgaben.
Zum Beispiel laß ich mir durch ein nicht gesetztes (gruenes) DTR anzeigen, daß mein RCX (LEGO) aktiv ist.

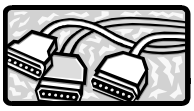
```
Word: LEDdwn
\ nur Kosmetik...
: LEDdwn ( -- )
    ModCtReg DUP P@ 1 OR
             SWAP P! ;
\ RCX antwortet nicht...

Word: LEDtrn
\ nur Kosmetik...
: LEDtrn ( -- )
    2 ModCtReg P! ;
\ auf Sendung

Word: LEDrcv
\ nur Kosmetik...
: LEDrcv ( -- )
    0 ModCtReg P! ;
\ auf Empfang

Word: 5DTR
: 5DTR ( -- )
    10 0 DO LEDtrn  100 MSEC
            LEDdwn  100 MSEC
       LOOP LEDtrn ;          \ Spieltrieb
```

Group: PortSteuerung

Die Portsteuerung enthaelt die zur Auswahl und Inititalisierung eines ComPorts benoetigten Worte.

```
Word: +DLAB
: +DLAB ( -- )
    LinCtReg DUP P@        \ Register auslesen
    %10000000 OR SWAP P! ; \ DLAB setzen,
                           \ zurueckschreiben

\ Wenn DLAB gesetzt ist, werden alle Schreib-
\ und Lesezugriffe auf ComPort 0 + nicht auf
\ das IOBufReg gelenkt, sondern auf LSBLtReg!
\ Wenn der darin enthaltene Teiler fuer die
\ Baudrate ueberschrieben wird, kann das die
\ Datenkommunikation stoeren.

Word: -DLAB
: -DLAB ( -- )
    LinCtReg DUP P@        \ Register auslesen
    %10000000 OR
    %10000000 XOR SWAP P! ;\ DLAB loeschen,
                           \ zurueckschreiben

\ Wenn DLAB gesetzt ist, werden alle Schreib-
\ und Lesezugriffe auf ComPort 0 + nicht auf
\ das IOBufReg gelenkt, sondern auf LSBLtReg!
\ Wenn der darin enthaltene Teiler fuer die
\ Baudrate ueberschrieben wird, kann das die
\ Datenkommunikation stoeren.

Word: <>BRK
\ Umschalter f r das Break-Bit
: <>BRK ( -- )
    LinCtReg P@ %01000000 XOR LinCtReg P! ;

\ Ausser in Test-Modi ist das BREAK immer
\ ausgeschaltet. Bei eingeschaltetem BREAK
\ koennen keine Daten ueber die Leitung gehen.

Word: NulParity
: NulParity ( -- )
    LinCtReg DUP P@ %00111000 OR
    \ Paritaetsbits loeschen
                    %00111000 XOR
            SWAP P! ;

Word: OddParity
: OddParity ( -- )
    LinCtReg DUP P@ %00111000 OR
                    %00111000 XOR
                    %00001000 OR
    \ ungerade Paritaet setzen
            SWAP P! ;

Word: EvnParity
: EvnParity ( -- )
    LinCtReg DUP P@ %00111000 OR
                    %00111000 XOR
                    %00011000 OR
    \ gerade Paritaet setzen
            SWAP P! ;

Word: 8Data
: 8Data ( -- )
    LinCtReg DUP P@ %00000011 OR
    \ "8 DatenBit" setzen
            SWAP P! ;
```

```
Word: 7Data
: 7Data ( -- )
    LinCtReg DUP P@ %00000011 OR
    \ "8 DatenBit" setzen
                    %00000011 XOR \ loeschen
                    %00000010 OR
    \ "7 Datenbit" setzen
            SWAP P! ;

Word: 1Stop
: 1Stop ( b -- b' )
    LinCtReg DUP P@ %00000100 OR
                    %00000100 XOR
    \ geloescht = 1 StopBit
            SWAP P! ;

Word: 2Stop
: 2Stop ( b -- b' )
    LinCtReg DUP P@ %00000100 OR
    \ gesetzt = 2 StopBit
            SWAP P! ;

Word: Baud
\ 'n' muss eine gueltige Baudrate sein!
: Baud ( n -- )
     115200.        \ Bezugsfrequenz (115.200
                    \ Bit/s) dividiert durch
     ROT UM/        \ gewuenschte Frequenz ist
                    \ die kodierte Baudrate.
     256 /MOD       \ Das Ergebnis wird in ein
                    \ hoeher- und ein nieder-
                    \ wertiges Byte aufgeteilt.
     +DLAB          \ Zugriff auf die
                    \ Latchregister schaffen
       MSBLtReg P!  \ hoeherwertiges Byte
                    \ schreiben
       LSBLtReg P!  \ niederwertiges Byte
                    \ schreiben
     -DLAB ;        \ Zugriffe wieder auf die
                    \ Steuerregister setzen

Word: ComInit
: ComInit ( -- )
    _com2 IS ComPort        \ Bei mir arbeitet
                            \ der RCX an COM2,
    OddParity  8Data  1Stop \ mit ungerader
                            \ Paritaet
    2400 Baud               \ sowie 2.400 BAUD,
                            \ 8 Databits und
    TRUE IS init?           \ 1 Stopbit
    LEDtrn ;

Word: InitCom
: InitCom ( -- )
    ComInit        \ Damit ich sehe, dass COM
                   \ initialisiert ist;
    5DTR           \ mit blinkendem DTR-Signal
                   \ am RS232-Tester
    LEDtrn ;


Group: Monitore

...wird bei Bedarf schrittweise ergaenzt...

Word: .RegMon
: .RegMon ( -- )
    26  1 XY ." UART 8250 - Register-Monitor"
    26  2 XY ." ---------------------------"
    31  4 XY ." LinCtReg " LinCtReg P@ b.
               5 SPACES
            ." ModCtReg " ModCtReg P@ b.
    31  6 XY ." SerStReg " SerStReg P@ b.
               5 SPACES
            ." ModStReg " ModStReg P@ b.
    31  8 XY ." IntAkReg " IntAkReg P@ b.
               5 SPACES
            ." IntIDReg " IntIdReg P@ b.
    +DLAB
    31 10 XY ." LSBLtReg " LSBLtReg P@ b.
               5 SPACES
            ." MSBLtReg " MSBLtReg P@ b.
    -DLAB ;


Group: Receiver

Der Empfaenger liest alle Byte, die bis zur Ueberschreitung einer voreingestellten Zeit im UART ankommen, in seinen Puffer. Die Zeiteinstellung kann im Wort "_rcv" veraendert werden. Der voreingestellte Wert entspricht ca. 275 mSek.

Word: testRxRD
\ Der Receiver geht davon aus, dass vor dem
\ Beschreiben des Puffers rcvdata der Puffer
\ geloescht und die Pointer reinitialisiert
\ wurden.
: testRxRD ( adr -- )
    SerStReg P@
    %11111110 OR       \ wenn das ausmaskierte
                       \ Bit gesetzt ist,
    %11111110 XOR      \ dann liegt ein Byte
        \   ^ RxRD     \ zum Abholen im
  ;                    \ Empfangsregister
                       \ iobufreg bereit

Word: getByte
\ Highlevel Forth reicht. Bei 2400 BAUD kommen
\ < 270 Byte/Sekunde in den UART. Das ist Zeit
\ genug, alle Byte aus dem UART abzuholen...
: getByte ( adr -- adr b )
    DUP DUP 4 + DUP @ 1+ -ROT
    DUP @ 1+ -ROT DUP -ROT @ SWAP 2055 +
    < IF !
         iobufreg P@ SWAP C!
    ELSE iobufreg P@
         2DROP 2DROP
    THEN ;

Word: _rcv
\ Der Receiver geht davon aus, dass vor dem
\ Beschreiben des Puffers rcvdata der Puffer
\ geloescht und die Pointer reinitialisiert
\ wurden.
: _rcv ( adr -- )
    NOW BEGIN                \ .RegMon
          testRxRD IF getByte
                      NOW    \ Timeout
                             \ zuruecksetzen
                   THEN
           5 TICKS?          \ ca. 0,275 Sekunden
        UNTIL                \ Feierabend machen
    DROP ;

Group: Transmitter

Hier wird die eigentliche Sendearbeit definiert. Der Transmitter arbeitet zeichenweise und kontrolliert vor jeder Aktion, ob das Senderhalte-Register des UART leer und der UART zur Aufnahme eines neuen Byte bereit ist.
```

UART-Controller für HOLON-Forth

```
Der Transmitter uebergibt den kompleten Inhalt
des Puffers trndata an den UART.

Word: testInit
\ Sendearbeit mit nicht initialisierten
\ Schnittstellen fuehren zu schwer nachvoll-
\ ziehbaren Fehlern.
: testInit ( -- f )    \ "adr" ist StartData
                       \ des urtctrl-Objektes
    Init?
    NOT IF InitCom
      THEN ;

Word: testTBE
: testTBE ( -- f )
    SerStReg P@         \ wenn das ausmaskierte
                        \ Bit gesetzt bleibt,
      %01011111 OR      \ dann ist das Sender-
                        \ Halteregister
      %01011111 XOR     \ bereit, ein neues
     \   ^TBE ;         \ Byte aufzunehmen

Word: sendByte
: sendByte ( adr -- adr )
    DUP DUP 2+ @ SWAP 1031 +
    < IF DUP 2+ DUP @ 1+ DUP ROT !
         C@ iobufreg P!
    THEN ;

Word: testEnde
: testEnde ( adr -- adr f )
    DUP DUP @ SWAP 2+ @ = \ wenn trnout = trnin
  ;                       \ dann ist trnout
                          \ fertig

Word: _trn
\ Der Transmitter geht davon aus, dass vor dem
\ Beschreiben des Puffers trndata der Puffer
\ geloescht und die Pointer reinitialisiert
\ wurden.
: _trn ( adr -- )    \ "adr" ist StartData des
                     \ uartctrl-Objektes
    testInit
    NOW BEGIN        \ .RegMon
          testTBE IF sendByte
                     testEnde IF DROP EXIT
                                 \ DROP adr
                             THEN
                THEN
        90 TICKS?     \ in 5 Sek. (90/18)
                      \ muessen sogar 1024
        UNTIL ;       \ Byte gesendet sein.



Group: Methods

Methoden   ("arbeitende"   Worte)   des   UART-
Controllers

Word: _auto
\ Das interne Parsing des UART-Controllers
: _auto ( adr -- )
    \ DROP            ...die Adresse StartData
    \ bleibt auf dem Stack
  ;

\ z.Zt. passiert noch gar nichts. Aber ab hier
\ koennen Worte definiert werden, die arbeiten
\ sollen, wenn der Controller ohne jegliche
\ Message aufgerufen wird.
```

```
Word: _clrtrn
\ Clear TransmitBuffer und -zeiger
: _clrtrn ( adr -- )
    DUP DUP 7 + SWAP !
    DUP DUP 7 + SWAP 2 + !
            8 + 1024 ERASE ;

\ Der Bereich trndata ist mit Nullen ueber-
\ schrieben, die Zeiger inptr und outptr zeigen
\ VOR das erste Byte in trndata.

Word: _clrrcv
\ Clear TransmitBuffer und -zeiger
: _clrrcv ( adr -- )
    DUP DUP 1031 + SWAP 4 + !
    DUP DUP 1031 + SWAP 6 + !
            1032 + 1024 ERASE ;

\ Der Bereich trndata ist mit Nullen ueber-
\ schrieben, die Zeiger inptr und outptr zeigen
\ VOR das erste Byte in trndata.

Word: _>trn
\ Ein Byte in den Transmitbuffer schreiben, an
\ die Stelle, die vom trnptr bezeichnet wird;
\ trnptr inkrementieren (siehe Dokumentation)
: _>trn ( b adr -- )
    DUP @ 2DUP SWAP 1031 +
    \ wenn die Bereichsgrenze trndata nicht
    < IF 1+ DUP ROT ! C!
    \ ueberschritten ist, dann trnin inkr.,
    ELSE 2DROP 2DROP
    \ und das Byte an die durch trnin
    THEN ;
    \ beschriebene Speicherzelle schreiben

\ Es werden maximal 1024 Byte akzeptiert. Was
\ darueber hinaus kommt, wird einfach
\ ignoriert.

Word: _rcv>
\ Ein Byte aus dem ReceiveBuffer herausholen
\ und auf den Stack legen
: _rcv> ( adr -- b )
    DUP 6 + DUP @ 1+ -ROT
  \ inkremeniert den Zeiger rcvout nur,
  \ wenn die Bereichsgrenze nicht erreicht ist
    DUP @ 1+ -ROT
    DUP -ROT @ SWAP 2055 +
    < IF !
    ELSE 2DROP
    THEN C@ ;
\ Wenn mehr als 1024 Byte aus dem Puffer
\ angefordert werden, wird immer nur das letzte
\ Byte wiederholt.

Word: _#trn
\ gibt die Anzahl der in den Puffer trndata
\ geschriebenen Byte zurueck;
\ das sollte gleich der Anzahl der an den UART
\ uebergebenen Byte sein
: _#trn ( adr - b )
    DUP @ SWAP - 7 – ;

Word: _#rcv
\ gibt die Anzahl der vom UART uebernommenen
\ Byte zurueck
: _#rcv ( adr - b )
    DUP 4 + @ SWAP 1032 + - 1+ ;
```

```
Group: Messages

Messages, die an den UART-Controller uebergeben
werden koennen

Word: parse
MESSAGE: parse ( ??? )
\ Startet ein internes Objektparsing und wird
\ als erste Message automatisch aufgerufen,
\ wenn keine andere Message an das Objekt
\ uebergeben wird

Word: clrtrn
MESSAGE: clrtrn ( -- )
\ Loescht den TransmitBuffer und den Zeiger in
\ diesen Puffer

Word: clrrcv
MESSAGE: clrrcv ( -- )
\ Loescht den ReceiveBuffer und den Zeiger in
\ diesen Puffer

Word: getdtadr
MESSAGE: getdtadr
\ Legt die Startadresse des Datenbereichs auf
\ den TOS

Word: >trn
MESSAGE: >trn
\ legt das Byte auf dem TOS in die Adresse
\ trndata + trnin; inkrementiert den trnin.

Word: rcv>
MESSAGE: rcv>

\ legt das Byte aus der Adresse
\ rcvdata + rcvout auf den TOS
\ inkrementiert den rcvout.

Word: transmit
MESSAGE: transmit
\ Sendebefehl; der Inhalt von trndata wird in
\ den UART gepumpt...


Word: receive
MESSAGE: receive
\ Empfangsbefehl; der UART wird solange
\ abgefragt, bis nichts mehr kommt. Die
\ empfangenen Byte werden im ReceiveBuffer
\ abgelegt.


Word: #trn
MESSAGE: #trn
\ gibt die Anzahl der gesendeten Byte zurueck

Word: #rcv
MESSAGE: #rcv
\ gibt die Anzahl der empfangenen Byte zurueck



Group: UARTCtrl

Hier ist der eigentliche Controller definiert;
ein HOLON-Objekt, das dem Anwender einen Trans-
mitter-Puffer und einen Receiver-Puffer zur
Verfuegung stellt, sowie Methoden zum Umgang
mit diesem Objekt.
```

```
Word: UART-Datenbereich

Text: Aufbau des UART-Datenbereiches
Adressen:

+--------------------------------------   StartData
v
00 00 00 00 0000....0000 0000....0000
|  |  |  |  |            +--------------
|  |  |  |  |             rcvdata = StartData + 1032
|  |  |  |  +---------------------------
|  |  |  |   trndata = StartData +    8
|  |  |  +-----------------------------
|  |  |   rcvout  = StartData +    6
|  |  +-------------------------------
|  |   rcvin   = StartData +    4
|  +---------------------------------
|   trnout  = StartData +    2
+-------------------------------------
 trnin   = StartData

Word: _uartctrl
\ Das Definitionswort fuer Objekte vom Typ
\ _uartctrl
ObjectType: _uartctrl
      Data: 2056 ALLOT
   Methods: _auto _clrtrn _clrrcv NOOP _>trn
            _rcv> _trn _rcv  _#trn   _#rcv
  Messages: parse clrtrn  clrrcv  getdtadr >trn
            rcv>  transmit receive #trn  #rcv

Word: UartCtrl
\ Der UART-Controller
\ _uartctrl UartCtrl

\ Dieser hier wird nicht gebraucht. Definieren
\ Sie sich selbst einen, fuer die Arbeit mit
\ LEGOs RCX vielleicht einen UART-Controller
\ mit dem Namen RCX ?!?
```

Neues vom Forth-Büro

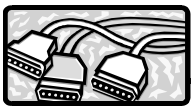
Liebes Mitglied, lieber Abonnent,

denken Sie bitte als Überweiser daran, Ihren Beitrag bis zum 15. März 2004 zu entrichten. Den Überweisungsvordruck haben Sie mit Post vom 15. Dezember 2003 zusammen mit der VD erhalten.

Wenn wir von Ihnen eine Einzugsermächtigung haben, so teilen Sie uns bitte Änderungen der Bankverbindung mit. Das erspart Ihnen und uns Kosten. Sie ersehen aus dem Beitragsbrief, den Sie mit Post vom 15. Dezember 2003 zusammen mit der VD erhalten haben, was wir gespeichert haben.
Ihr Beitrag wird am 15. März 2004 eingezogen.

Herzlichen Gruß, Ihr Forth-Büro

Rolf Schöne



UART-Controller für HOLON-Forth

Definitionen zur Portsteuerung

1024	400h	buflen	Integer		Basiswort zur Arbeit mit der Pufferlaenge
		trnbuf	Feld	1024 Byte	TransmitPuffer
		rcvbuf	Feld	1024 Byte	ReceivePuffer
	0	trnptr	Integer		Pointer in den trnbuf
	0	rcvptr	Integer		Pointer in den rcvbuf
	0	tmpptr	Integer		Pointer in diverse Felder (temporaere Nutzung)
	x	_com1	Integer		Basisadressen der vier moeglichen COM-Ports
	x	_com2	Integer		Die tatsaechlichen Adressen werden waehrend der
	x	_com3	Integer		Kompilatioon aus dem BIOS ausgelesen
	x	_com4	Integer		
	x	ComPort	Integer		Adresse des aktiven COM-Ports
	x	IOBufReg	Definition		Adresse des aktiven COM-Ports Ein- und Ausgaberegister
	x	INTAKReg	Definition		Adresse des aktiven COM-Ports + 1 Interrupt-Aktivierungsregister
	x	INTIDReg	Definition		Adresse des aktiven COM-Ports + 2 Interrupt-Identifizierungsregister
	x	LinCtReg	Definition		Adresse des aktiven COM-Ports + 3 Line-Controlregister
	x	ModCtReg	Definition		Adresse des aktiven COM-Ports + 4 Modem-Controlregister
	x	SerStReg	Definition		Adresse des aktiven COM-Ports + 5 Serielles-Statusregister
	x	ModStReg	Definition		Adresse des aktiven COM-Ports + 6 Modem-Statusregister
	x	LSBLtReg	Definition		Adresse des aktiven COM-Ports nur bei gesetztem DLAB in LinCtReg niederwertiger Teil der Baudrate
	x	MSBLtReg	Definition		Adresse des aktiven COM-Ports +1 nur bei gesetztem DLAB in LinCtReg hoeherwertiger Teil der Baudrate

Objekt UART-COntroller

Messages:		Methoden:	
vordefiniert	neu		
	parse	_auto	internes Parsing (aktuell nicht "belegt")
	clrtrn	_clrtrn	clear TransmitBuffer, TransmitPointer
	clrrcv	_clrrcv	clear ReceiveBuffer, ReceivePointer
	getdtadr	NOOP	StartAdresse DATA zum TOS
	>trn	_>trn	fuegt das Byte auf dem TOS in den Transmitpuffer, inkremenitert den Transmitter-Zeiger
	rcv>	_rcv>	holt ein Byte aus dem Receivepuffer auf den TOS, inkrementiert den Receivepuffer
	transmit	_trn	uebergibt den Inhalt trnbuf an den UART
	receive	_rcv	holt Daten vom UART und legt diese in den ReceivePuffer
	#trn	_#trn	gibt die Anzahl der gesendeten Byte zurueck
	#rcv	_#rcv	gibt die Anzahl der empfangenen Byte zurueck

Bedienungsbeispiele fuer das Objekt UARTCtrl:

	clrtrn	uartctrl -	loescht den Transmitterpuffer
	getdtadr	uartctrl -	legt die Adresse des Datenbereiches auf den TOS (vielleicht braucht man das ja mal)
$FF	>trn	uartctrl -	legt das Byte in den naechsten freien Platz im TransmitterPuffer; es koennen maximal 1024 Byte geschrieben werden. Werden mehr Byte geschrieben, ueberschreibt das Objekt immer das letzte Byte.
	rcv>	uartctrl -	holt das naechste Byte aus dem Empfaengerpuffer auf den TOS
	transmit	uartctrl -	sendet die Byte im Transmitterpuffer an den UART. Die Transmission erfolgt gepuffert. Das jeweils naechste Byte wird erst dann an den UART gegeben, wenn das Sendehalteregister leer ist.
	receive	uartctrl -	Es werden solange vom UART Byte abgefragt, bis mindestens 0,275 Sekunden lang keine neuen Byte mehr im Empfangsregister stehen.
	#trn	uartctrl -	gibt die Anzahl der gesendeten Byte auf den TOS
	#rcv	uartctrl -	gibt die Anzahl der empfangen Byte auf den TOS

Um "von aussen" Abfragen auf die Pufferinhalte durchzufuehren, z.B. um die Antworten des RCX auf richtige Ausfuehrung abzugleichen, lassen sich mit #trn, #rcv und getdtadr alle benoetigten Worte nach Belieben definieren. Braucht die Schnittstelle weitere Funktionen?

Eine virtuelle nicht-deterministische Maschine in Forth

James A. Boyd

(Forthwrite 123, Dez. 2003)
(Mit freundlicher Genehmigung
der Redaktion der Forthwrite)

Im ersten Teil dieses Beitrages beschreibt J. Boyd eine Technik, die in Applikationen vom Typ „suchen“ verwendet werden kann. Der abschließende Teil soll in der nächsten Ausgabe *(der Forthwrite)* erscheinen.

Eine nicht-deterministische Maschine ist eine faszinierende theoretische Konstruktion. Sie hat die Fähigkeit, bei einem gegebenen Problem, falls es überhaupt eine Lösung hat, den richtigen Lösungswert zu finden. Algorithmen für nicht-deterministische Maschinen können beträchtlich schneller sein, als die für deterministische. Dieser Beitrag beschreibt, wie eine virtuelle nicht-deterministische Maschine implementiert werden kann; es ist ein Experiment in Forth – das hoffentlich einiges vom Potential der theoretischen Maschine verwirklicht. Diese Arbeit wurde angeregt von der Diskussion zwischen Ellis Horowitz und Sartaj Sahni über nicht-deterministische Maschinen und einem Beitrag von L. L. Odette.

Der theoretische Rahmen

Horowitz und Sahni brachten das Konzept eines nicht-deterministischen Algorithmus in die Diskussion von „NP-hard und NP-vollständig“-Problemen ein. Sie stellten eine Maschine dafür vor, die aber nur in der Theorie existierte, lieferten damit aber starke intuitive Gründe für den Schluss, daß selbst schnelle deterministische Algorithmen Aufgaben aus dem Bereich NP nicht wirklich lösen können, weil sie extrem viel Zeit dafür verbrauchen.

Viele Aufgaben der Computerwissenschaft haben inzwischen gut bekannte Algorithmen, deren Rechenzeiten sich in zwei Clustern häufen. Die Rechenzeiten aus der ersten Gruppe sind durch Polynome niedrigen Grades beschreibbar. Man nennt sie „in P“ oder zugehörig zu der Gruppe aller Aufgaben, die in polynominaler Zeit lösbar sind. Von diesen Aufgaben wird angenommen, daß sie behandelbar sind, also lösbar innerhalb einer angemessenen Frist, selbst für sehr große Aufgabenstellungen. Zu solchen Aufgaben rechnet man das Multiplizieren von Matrizen, das Durchsuchen geordneter Datenmengen, Polynome lösen und Daten sortieren.

Die Aufgaben der anderen Gruppe und ihre besten Algorithmen haben hingegen Lösungszeiten, die selbst durch größte Polynome nicht mehr eingegrenzt werden können. Sie werden nichtpolynominale Aufgaben (NP-Probleme) genannt. Von diesen NP-Problemen heißt es, sie seien von Computern unbehandelbar, typischerweise behaftet mit exponentiell ansteigenden Lösungszeiten. Obwohl also theoretisch lösbar, wächst die Zeit für deren Lösung so schnell mit dem Aufgabenumfang, dass sie nicht mehr wirklich durchführbar sind, wenn mittlere oder große Datenmengen anfallen.

Aber es gibt eine Unterklasse an NP-Problemen, die „NP-vollständig“ genannt werden. Alle Aufgaben in dieser Klasse sind theoretisch vollständig lösbar. Und falls es überhaupt irgendein NP-Problem geben sollte, das in polynominaler Frist lösbar ist, dann müssen auch alle NP-vollständig Probleme in polynominaler Frist lösbar sein. Bis jetzt war aber noch niemand in der Lage, überhaupt irgendein NP-Problem in einen deterministisch polynominalen Algorithmus zu fassen. Als Beispiel solcher Probleme nenne ich mal das Problem des Handelsreisenden, das Rucksackproblem, den Hamiltonkreis, und vermutlich gehört auch das N-Dame-Problem dazu.

Die nicht-deterministische Maschine kann nun, theoretisch, für ein NP-vollständig Problem eine Lösung in polynominaler Frist finden. Ein nicht-deterministischer Algorithmus für ein NP-vollständig Problem ist leichter zu beschreiben als sein deterministisches Gegenstück. Und die Rechenzeit für den nicht-deterministischen Algorithmus liegt normalerweise wieder in den Grenzen eines Polynoms geringen Grades. So eine nicht-deterministische Maschine hat die Operatoren `choice`, `failure` und `success`. Die virtuelle nicht-deterministische Maschine hat die gleichen drei Operatoren. In der Datei **VNM.F** steht der Code für diese virtuelle nicht-deterministische Maschine.
[*Anmerkung: der Code wird voraussichtlich in der VD 02/2004 abgedruckt werden, fep*]
Aus einem gegebenen Bereich wählt `choice` zufällig eine Zahl so aus, daß `success` ausgeführt und `failure` vermieden wird, falls es so eine Zahl überhaupt gibt.

- `Choice` entnimmt den Wert **n** vom Stack und legt dafür einen Wert aus dem Bereich Null bis **n** dort ab.
- Der Ausdruck `over - choice +` nimmt die zwei Werte **lo** und **hi** vom Stack und legt dafür einen Wert aus dem Bereich **lo** und **hi** inklusive der Grenzen dort ab.

Der Ausdruck `failure` wird hinter den Test gesetzt, den `choice` durchführt, weshalb `failure` nur ausgeführt werden kann, wenn der Test nicht bestanden worden ist. Im folgenden Codefragment wird gezeigt, wie Aktionen noch nach `failure` ausgeführt werden können:

- ... if failure word1 ... wordn then ...

`word1` bis `wordn` werden also nur ausgeführt, wenn auch `failure` durchgeführt worden ist. Sie können dann alles mögliche machen, den Stack leeren, eine Fehlermeldung geben, oder den Algorithmus verlassen. Beachten Sie aber, dass diese Worte hinter `failure` und innerhalb derselben `IF ...`

THEN Phrase stehen.

Das Wort success hingegen wird an das Ende eines Algorithmus gesetzt, da es nur ausgeführt wird, falls die Aufgabe lösbar war. Der eine Fall, das choice ein failure nicht umgehen konnte obwohl es doch eine Lösung gab, tritt nur dann auf , wenn success vor failure ausgeführt wurde. Das nicht-deterministische Verhalten von choice, seine Fähigkeit, eine Wahl zu treffen, mit der failure umgangen werden kann, ist auf den Codeabschnitt zwischen sich selbst und dem ersten Auftreten von success oder failure beschränkt. Deshalb steht success normalerweise am Ende des Algorithmus.
Um einen nicht-deterministischen Algorithmus zu schreiben, wird dem Wort choice der Bereich der möglichen Werte für jedes Element des „Lösungsvektors“ vorgegeben, und dann, je nach Ergebnis der Wahl von choice, das Wort failure entweder vermieden oder ausgeführt. Das Verhalten eines nicht-deterministischen Algorithmus ist dann festgelegt durch den Bereich der möglichen Werte für jedes Element aus dem Lösungsvektor und den Tests, oder den Prädikaten, wie Odette die Tests nannte. Überlegungen zum Backtracking, oder dem, was zu tun ist, wenn ein Element den Test nicht besteht, behindern die Überlegungen, einen nicht-deterministischen Algorithmus zu schreiben, nicht. Falls irgendeine Auswahl dazu geführt hat, daß failure ausgeführt worden ist, dann gibt es keine Lösung.
Im Gegensatz zur theoretischen Maschine von Horowitz und Sahni beenden die Worte success und failure das Programm nicht, sondern nur das nicht-deterministische Verhalten des Algorithmus. Sie können in anderen Worten stecken, die vom Hauptalgorithmus aufgerufen werden, wie im Falle von Prädikaten. Aber diese beiden Worte sind nur in nicht-deterministischen Algorithmen brauchbar. Und weil success und failure das nicht-deterministische Verhalten des Algorithmus beenden, kann success auch am Anfang eines deterministischen Teiles stehen, um diesen abzuschirmen von einem nicht-deterministischen, welcher nicht ordentlich mit success und failure abgeschlossen werden konnte.

Das n-Damen-Problem

Nehmen wir als Beispiel zunächst mal das Acht-Damen-Problem. Das Ziel ist es, acht Damen auf dem regulären Schachbrett so aufzustellen, dass keine Dame eine andere bedroht. In der Datei NQUEENS.F ist der Code für nqueens abgelegt, dem nicht-deterministischen Algorithmus für das n-Damen-Problem. Wie Sie sehen, steht success am Anfang und am Ende des Algorithmus, während choice und failure im Unterprogramm (nqueens) stehen, das die eigentliche Arbeit erledigt. Wir sehen auch, das der Prädikator attacks? gleich hinter choice kommt, um dessen Wahl zu bestätigen. Das Bild1 zeigt eine Lösung des Acht-Dame-Problems. Die Lösung ist in Form einer Liste von Positionen in der Reihe wiedergegeben: 3 6 2 5 8 1 7 4. Ein konventioneller Algorithmus hätte gerade **diese** Liste nicht als erstes ausgegeben. Aber ein nicht-deterministischer Algorithmus gibt von den 92 Möglichkeiten für das Acht-Dame-Problems **irgendeine** als Lösung aus.

Ein konventioneller Algorithmus für das Backtracking muss jeden Wert ausprobieren, der für jedes Element der Aufgabe existiert, bis eine unbedrohte Position für die Dame gefunden ist. Falls dabei keine solche Position gefunden wird, muss zurückgegangen werden (backtrack) zur vorherigen Position, um von dort die nächste Position zu prüfen. Aber wenn auch die vorherige Dame dann nicht mehr sicher ist, muss auch diese zurückverfolgt werden. All dieses Backtracking verlangsamt den Algorithmus erheblich. Das Acht-Dame-Problem braucht 876 Versuche, wenn man unter einem Versuch jede neuen Position für die Dame einschließlich Test für diese Position versteht.
Mit der Fähigkeit der nicht-deterministischen Maschine, jeweils den richtigen Wert für ein Element des Lösungskonzeptes auszuwählen, bedarf es keines Backtrackings mehr. Für das Acht-Damen-Problem braucht die nicht-deterministische Maschine auch nur acht Versuche, um alle Damen richtig zu platzieren.

Das Acht-Damen-Problem kann generalisiert werden zum n-Damen-Problem. Dabei ist es das Ziel, **n** Damen auf einem **n** mal **n** großen Schachbrett zu platzieren. Die nicht-deterministische Maschine braucht dazu auch nur **n** Versuche, um alle **n** Damen zu positionieren. Und da es keine Lösung für ein 2 mal 2 Damen und 3 mal 3 Damen Problem gibt, würde der nicht-deterministische Algorithmus mit einer Fehlermeldung abschließen, und die Aufstellung würde bedeutungslos.

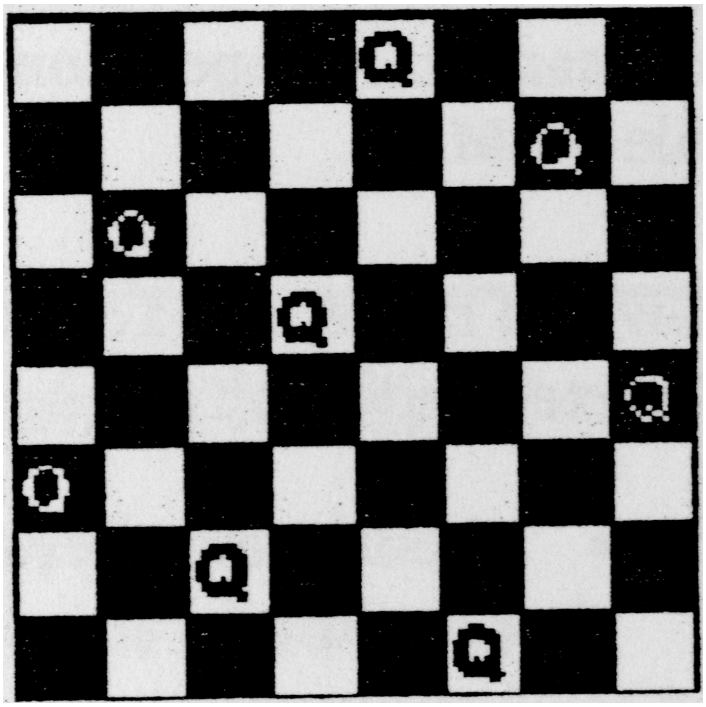


Bild 1: Eine Lösung des Acht-Damen-Problems

Der Beitrag von *James A.Boyd* wurde übersetzt von *Michael Kalus* und kommentiert von *Bernd Paysan* und *Fred Behringer*.

Eine nicht-deterministische Maschine

Eine nicht-deterministische Maschine, was ist das?

Ganz einfach - so eine Maschine hat die Fähigkeit, die richtige Auswahl für ein Problem zutreffend zu erraten, falls es so eine Auswahl überhaupt gibt. Ein Algorithmus heißt nicht-deterministisch, wenn er nicht-deterministische Operationen auf einer nicht-deterministischen Maschine verwendet. Nehmen wir mal an, das Wort `?even` hinterließe ein TRUE als Flag, wenn eine gerade Zahl auf dem Stack liegt. Im Wort `evenDemo` bewirkt `choice` nun, dass immer irgendeine gerade Zahl aus dem Bereich Null bis **n** auf dem Stack liegt.

```
: evenDemo   (n -- )
  choicedup ?even
  if  success  .
  else  failure drop
  then ;
```

Und das Wort `oddDemo` wählt mittels `choice` immer eine ungerade Zahl aus dem Bereich Null bis **n** wenn **n** > 0 ist.

```
: oddDemo   (n -- )
  choicedup ?even
  if  failure drop
  else  success  .
  then ;
```

James A. Boyd über sich:

Ich bin Christ, Ehemann und Vater von vier Kindern. Und obwohl ich kein Berufsprogrammierer bin, habe ich in Forth seit November 1991 programmiert. Mein Interesse an nicht-deterministischen Maschinen wurde von L.L. Odette geweckt, durch seinen Artikel in Dr. Dobbs Journal und sein Buch mit dem Titel „Grundlagen der Computer Algorithmen“. Im Kapitel über NP-hard und NP-vollständig Probleme beschrieben Ellis Horowitz und Sartaj Sahni eine theoretische Maschine, die nicht-deterministisch genannt wurde.

Der Übersetzer (**Michael Kalus**) und der Editor (**Friederich Prinz**) haben bei der ersten Durchsicht des Artikels aus der Forthwrite ihren Augen nicht getraut. Eine nicht-deterministische Maschine? Programmiert in Forth? Den Praktikern gellen die Alarmglocken ohrenbetäubend. Wir haben Rat gebraucht und bekommen. Zunächst ein Auszug aus einer E-Mail, in der *Fred Behringer* verschiedene Details zu Inhalten des Artikels anschaulich erklärt:

....
Ein Hamilton-Kreis ist ein Begriff aus der Graphentheorie. Ein Rundreiseproblem (jede Stadt soll ein und nur einmal besucht werden, und das auf einem minimalen Weg) braucht keine Lösung zu haben. Es hat eine Lösung, wenn der zugeordnete (gerichtete) Graph (eine irreflexive Relation) einen Hamilton-Kreis hat.

Beim Rucksackproblem geht es darum, Gegenstände (aus einer gegebenen Menge solcher) so auszuwählen, dass sie insgesamt noch in den Rucksack passen, dass dabei aber die Summe ihrer "Nützlichkeitswerte" maximiert wird.
Auf gegenseitige Abhängigkeiten wird beim "einfachen" Problem keine Rücksicht genommen: Der Nützlichkeitswert eines Flaschenöffners hängt natürlich davon ab, ob ich wenigstens eine klitzekleine Flasche Bier mitnehme. Der Nützlichkeitswert einer Flasche Bier (in Normalausführung) hängt wiederum davon ab, ob ich einen Flaschenöffner in den Rucksack gepackt habe.
(Man wundert sich, was einem alles einfällt, wenn man auf einem längeren abendlichen Spaziergang versucht, eine (im Rucksack) mitgenommene Flasche ohne Öffner zu öffnen.)
Die Mathematiker idealisieren. (Es besteht kein Grund, sie deshalb auszulachen. Es ist die einzige Möglichkeit, als Spezies zu überleben.) Der Praktiker würde sofort erkennen, dass es einen wesentlichen Unterschied ausmacht, ob er sich mit der Flasche ohne Öffner im Wald befindet oder in der S-Bahn oder im Stadtgebiet - unbeobachtet oder von Passanten umgeben.
....

Der Informatiker – und damit Fachmann – *Bernd Paysan* schreibt:

Ja, ob man theoretische Informatiker nun ernst nehmen kann oder ob das alles Unfug ist, was sie sagen, das ist eine schwierige Frage. Schließlich ist das alles mehr oder weniger Mathematik, und die Voraussetzungen, die man da macht, sind erst mal willkürlich und nicht notwendigerweise in unserer Welt implementierbar. Die Folgerungen sind nichtdestotrotz immer richtig.

Der "Trick" an der Maschine ist eigentlich ganz einfach: Die Rate-Funktion "choice", die immer ein "richtiges" Ergebnis liefert (wenn auch nicht immer dasselbe), ist in der Annahme O (1), also unabhängig von der tatsächlichen Komplexität des Problems. Realisiert man eine solche nicht-deterministische Maschine aber wie hier auf Basis einer deterministischen, verhält sie sich natürlich *nicht* so. Schließlich rät die deterministische ja nur, und muß dann mit failure und der gespeicherten History doch wieder ein Backtracking machen.

Nicht-deterministische Maschinen spielen bei den sogenannten "Quanten-Computern" eine wichtige Rolle. Die Hoffnung ist hier, daß die Quantenmechanik es ermöglicht, einen solchen Computer zu bauen. Der könnte dann schwierige Probleme, bei denen man viel ausprobieren muß, schnell lösen (Beispiel: Primfaktoren einer großen Zahl). Ob die Quantenmechanik das aber wirklich hinkriegt, das hängt davon ab, ob Bohm oder Bohr recht haben. Bohr ist der mit dem Kopenhagen-Modell (inherent indeterministisch, Wellenfunktionen überlagern sich, werden bei der Beobachtung festgelegt, d.h. die Wellenfunktion "kollabiert", mit der merkwürdigen Sonderstellung des "Beobachters"), Bohm dagegen behauptet, die Quantenmechanik hätte einen deterministischen Unterbau, den man nur nicht beobachten kann ("hidden variables").

Es gibt übrigens auch andere Berechnungs-Modelle, die NP-

Probleme in polynominaler Zeit lösen können, nämlich solche, bei der sich der Computer einfach bei jedem Probierschritt vervielfacht. Auch hierzu gibt es Realisierungen, nämlich mittels DNA (die ist klein genug, und kann das Vervielfachen). Interessanterweise gibt es auch da eine Quantenmechanik-Interpretation, die genau darauf hinausläuft, nämlich die Vielweltentheorie (wobei bei der das Reagenzglas nicht überläuft, wenn man zu viele Möglichkeiten ausprobieren muß ;-).
Als Informatiker freut man sich natürlich, daß auch die Physik letztendlich nach der Antwort auf die Frage P=NP sucht.

Liebe Leser,
Bitte schreiben Sie uns, ob und wie Ihnen dieser Exkurs in die theoretische Informatik zusagt. Wir wollen den Autor und die Forthwrite bitten, uns auch den zweiten Teil der Arbeit von James Boyd zum Abdruck zu überlassen. Vielleicht ist dies dem einen oder dem anderen Leser unserer Zeitschrift ein Ansporn, eigene Begegnungen mit der „grauen Theorie“ zu veröffentlichen. *fep*

Dosemu und Holon einrichten

Martin Bitter
Merhhoog

Stichworte: Holon, dosemu, Linux, RS232-Schnittstelle

Wie in dieser Ausgabe an anderer Stelle berichtet wird, hat Fritz Prinz seine Bemühungen um den Lego-RCX und das Holon von Wolf Wejgard neu aufleben lassen und schon erste ausgezeichnete Erfolge erzielt. An dem Vorhaben sind auch Wolf Wejgaard und ich beteiligt. Schon vor zwei Jahren hatte ich ein wenig mit Holon 'gespielt', habe es dann aber arg vernachlässigt – vor allem, weil ich zu Linux wechselte und die damalige Version eines DOS-Emulators nicht zufriedenstellend dazu überreden konnte, mit der RS232 Schnittstelle zusammenzuarbeiten. Das ist mir inzwischen gelungen. Um Anderen die Arbeit ein wenig zu erleichtern, hier meine Erfahrungen mit dem Linux-dosemu-Holon Triumvirat.

Dosemu: Wie der Name schon nahe legt, ist dosemu ein Programm, das unter Linux ein DOS emuliert. *dosemu* wird (nach erfolgreicher Installation) durch die Eingabe von dosemu an einer Linuxkonsole gestartet. (Linux bietet standardmäßig mehrere unabhängige Konsolen an, bei entsprechender Installation auch eine windowsähnliche Konsole namens X.) Mit dem Befehl *xdosemu* wird der Emulator in einem X-Fenster gestartet. Ich selbst starte die X-Version. Das hat mehrere Vorteile: 1. Es sind Grafikausgaben u.a. grafische Zeichensätze möglich 2. Ich kann nicht nur mehrere dosemuen starten, sondern zusätzlich ihre Fenster gleichzeitig sichtbar halten. Das letztere ist im Zusammenspiel mit Holon vorteilhaft.

Dosemu einrichten: Dosemu kann bei www.dosemu.org abgeholt werden. Will man dosemu Zugriff auf 'geschützte' Funktionen erlauben, dazu gehört eben auch der Zugriff auf den RS232-Port, so muss man systemweit installieren und in einem späteren Schritt den ausgewählten Usern die entsprechenden Rechte einräumen. Genaueres zur Installation steht in den entsprechenden READMEs, in meinem Fall in der Datei INSTALL[1)].

Das angeführte Script nimmt Ihnen bis auf das Downloaden der beiden /dosemu-1.1.99.1.tgz und dosemu-freedos-b9-bin.tgz alles Notwendige ab. Voraussetzung: Sie müssen die beiden Dateien in Ihrem Homeverzeichnis abgelegt haben und Sie müssen als User mit Superrechten eingetragen sein (Probe: Klappt bei Ihnen der Befehl 'sudo ls' ? Ja? Dann gelingt auch die Ausführung des Scriptes.)

Nach der hoffentlich erfolgreichen Installation findet man unter /etc/dosemu/ mehrere Dateien. In der Datei /etc/dosemu/dosemu.users steht, welchen Usern dosemu erlaubt, auf die geschützten Funktionen zuzugreifen. Ich habe dort nur eine Zeile stehen: martin c_all . martin ist mein Username und c_all stellt mir alle Möglichkeiten des dosemu zur Verfügung. In der Datei /etc/dosemu/dosemu.conf hat das Script noch die beiden Zeilen **$_com1 = "/dev/ttyS0"** und **$_com2 = "/dev/ttyS1"**. Es ist nicht notwendig, den Zugriff auf die dazugehörigen Portadressen zu erlauben. Dies wird vom dosemu automatisch erledigt. Der Zugriff von DOS aus durch direktes Schreiben und Lesen der Portadressen ist jetzt gestattet.

Der Zugriff auf die 'originale' DOS-Partition kann unterschiedlich realisiert werden. Ich habe einfach in meinem Homeverzeichnis einen Link auf die DOS-Partition eingerichtet, die bei mir auf dem Windowslaufwerk E: liegt (~/win-e --> / winplatten/win-e). Jetzt kann mit der Kommandozeile der dosemu gestartet werden.

xdosemu -s -home
xdosemu startet den Emulator in einem Fenster, **-s** erlaubt Rootrechte, das ist notwendig für den Zugriff auf die Comports und **-home** benutzt das eigene Heimatverzeichnis als D: unter DOS. Sie erinnern sich: Ich kann darüber auf meine DOS Partition zugreifen.

Holon ist ein dreigliedrig strukturiertes Forth, das mit zwei kommunizierenden Programmen arbeitet. (Eines, das eigentliche Holon (holon86.exe) stellt die Entwicklungsumgebung unter DOS dar.) Ein zweites Programm, der Monitor, läuft in der Zielumgebung. Das kann je nach Einstellung ein via RS232 angeschlossenes embedded System sein, z.B. ein 68HC11 oder ein zweiter DOS-PC (via RS232) oder ein Monitorprogramm auf dem gleichen Rechner oder eine so genannte Co-Routine.
Die Arbeit mit dem Monitorprogramm wmon86.exe gestaltet sich in etwa so: Auf dem Holon, das unter Windows – jetzt auch unter Linux – in einer DOS-Box läuft, wird Code entwik-

1) Ich setze voraus, das Linuxbenutzer das können. Bei Problemen steh ich gerne für Nachfragen zur Verfügung. Ich selbst bin immer noch blutiger Linuxanfänger – aber immerhin, bei mir ist es gelungen:-)

kelt und an den Monitor geschickt. Dort kann er, bei laufendem Holon, erprobt werden. Intern geschieht das, indem beide Programme auf eine (shared) Datei c:/holon.dde zugreifen. Läuft der Code zufriedenstellend, kann ein Turnkey erzeugt werden, das auf dem Zielsystem – hier ein DOS-PC – einsetzbar ist. Für den Fall, dass kein Betriebssystem (z.B. DOS selbst) zur Verfügung steht, auf dem zwei DOS-Boxen gleichzeitig betrieben werden können, kann als Monitor eine Coroutine eingesetzt werden. Das ist ein Programmstück, das unter Holon in einem reservierten Speicherbereich läuft und das mit Holon genauso kommuniziert, wie ein zweiter DOS-PC, bzw. wie wmon86.exe. Durch Tastendruck kann zwischen beiden (Holon und Coroutine) hin und her geschaltet werden und genauso ein Turnkey erzeugt werden.

Holon und dosemu: Ist dosemu einmal richtig installiert, kann Holon betrieben werden. Dabei gilt es in unserem Fall (COM-Ports) beim Start zu beachten, was geplant ist.

- Holon und wmon86. Wir wollen unter Holon auf den COM-Port zugreifen und zwar vom wmon86 aus. Würden wir Holon in einer Dos-Box (dosemu) mit Zugriffsrechten auf die COM-Ports starten, so würde Holon diese Ports blockieren! Deshalb starten wir als normaler User ohne Sonderrechte, d. h. ohne den Parameter **-s**. Das ist unsere Startzeile: **xdosemu -home** In dem nun gestarteten DOS-Fenster kann jetzt Holon gestartet werden. Bei mir also **d:** (enter) **cd 4th/holon/holon.rcx** (enter) **holon4de**[2)]. Holon läuft! Mit Shift-Strg-F3 öffnet sich ein Dialog in dem, falls notwendig, der Windowsmonitor (wmon86.exe) als Zielsystem eingestellt werden kann.
  Nun kann mit **xdosemu -home -s** ein DOS-Fenster für den Windowsmonitor (wmon86.exe) geöffnet werden. Hier ist der Parameter **-s** wichtig, denn wir wollen ja, dass der Monitor Zugriff auf die RS232 Schnittstelle hat. Manchmal wird der Windowsmonitor von Holon nicht sofort erkannt. Dann genügt es ihn zu beenden und neu zu starten.
- Holon mit Coroutine. In diesem Fall reicht es, das DOS-Fenster für Holon mit Superrechten zu starten: **xdosemu -home -s**. Da die Coroutine ja ein 'Teil' von Holon ist, haben beide jetzt Zugriff auf die COM-Schnittstelle. Nicht vergessen mit Shift-Strg-F3, die Coroutine als Zielsystem einzutragen.

Ups! Alle Funktionen von Holon lassen sich mit dosemu nutzen. Alle? Nein :-(Meine Tests haben ergeben, dass sich ganz unglücklicherweise mit Holon erarbeitete Module nicht auslagern (exportieren) lassen. Sobald ich Alt-F4 drückte (die Tastenkombination zum Exportieren) stürzte dosemu ab! Lange Zeit und intensive Beschäftigung mit der Einstellung des dosemu und seiner Tastaturbelegung halfen nicht. (Der ein oder andere Leser schmunzelt jetzt vielleicht schon.) Irgendwann kam ich darauf: Alt+F4 ist die Tastenkombination, die dem X-Manager sagt, das gerade geöffnete Fenster zu schließen. Trotz vielem Googeln fand ich für dieses Problem zwar keine Lösung[3)], aber per Zufall einen Ausweg. In göttlicher Voraussicht (oder durch Zufall?) hat Wolf Wejgaard gerade bei der Exportfunktion (Alt-F4) auch die Tastenkombination AltGr+F4 nutzbar gemacht. Jetzt ist Holon unter Linux uneingeschränkt nutzbar.

Zur Bequemlichkeit habe ich mir noch ein Script geschrieben, das mir zweimal dosemu startet, in die passenden Verzeichnisse wechselt und den Winmon startet (warum das mit dem Holon4de[2)] nicht klappt weiß ich nicht. Vielleicht einer der Leser? Jedenfalls startet holon4de in Zeile 2, aber es reagiert dann nicht mehr auf Tastenanschläge):

```
#!/bin/bash
xdosemu -home -E "d:|cd 4th\holon\holon.rcx" &
# xdosemu -home -E "d:|cd 4th\holon\holon.rcx|holon4de" &
xdosemu -s -home -E "d:|cd 4th\holon\holon.rcx\wmon86"
```

Verkettungen: Ich benutze Holon letztendlich, um mit Fritz Prinz und Wolf Wejgaard den Lego-RCX-Brick zu programmieren. Bei mir führt das zu folgender, funktionierender Verschachtelung: Linux --> dosemu --> Holon --> Windowsmonitor (RS232) --> Sendetower --> RCX.
Und das klappt!

Installationsscript für systemweiten dosemu mit freiem Zugriff auf COM-Ports.

```
#!/bin/bash
echo -n "Verzeichnis in dem die .tgz's abgelegt sind : " &&
read TGZ_DIR &&
cd /usr/src &&
tar -xzf $TGZ_DIR/dosemu-1.1.99.1.tgz &&
cp $TGZ_DIR/dosemu-freedos-b9-bin.tgz ./dosemu-1.1.99.1/
dosemu-freedos-bin.tgz &&
cd dosemu-1.1.99.1 &&
echo -e "Der dosemu wird compiliert. Das kann dauern.\nBei
Abbruch wegen Fehlern finden Sie naehere Informationen \
in der Datei:\n /usr/src/dosemu-1.1.99.1/MAKE.meldungen"
&&
make >MAKE.meldungen 2>>MAKE.meldungen &&
echo -e "dosemu erfolgreich Kompiliert. \nStarte jetzt die systemweite Installation ...\nBei Fehlern s.o." &&
make install >>MAKE.meldungen 2>>MAKE.meldungen &&
echo -e "dosemu erfolgreich installiert. \nStarte jetzt die Konfiguration (ready for Holon) ... "&&
cd /usr/src &&
echo -n "Installation fuer Benutzer : " &&
read INST_USER &&
echo "$INST_USER c_all" > /etc/dosemu/dosemu.users &&
echo "\$_com1 = \"/dev/ttyS0\"" >> /etc/dosemu/dosemu.conf
&&
echo "\$_com2 = \"/dev/ttyS1\"" >> /etc/dosemu/dosemu.conf
&&
rm -fR dosemu-1.1.99.1 &&
echo "Fertig!" &&
echo "Bitte als normaler Benutzer (su INST_USER)'xdosemu -
s -home' eingeben!"
```

2) Holon4de ist eine von Fritz Prinz eingedeutschte Holonversion. Er stellt sie auf Anfrage gerne zur Verfügung.
3) Vielleicht ist jemand unter den Lesern, der helfen kann. Hier die notwendigen Daten: Redhat 9.0, Gnome2

Gehaltvolles

zusammengestellt und übertragen

von
Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 123 Dezember 2003

1 Editorial
Graeme Dunbar <g.r.a.dunbar@rgu.ac.uk>

Die Zukunft der FIGUK-Zeitschrift Forthwrite steht auf dem Spiel, sagt Graeme, wenn die Mitglieder keine Artikel einsenden. Das war auch der Tenor der Mitgliederversammlung Ende Oktober 2003. Wenn schon keine Artikel, dann wenigstens "Briefe an die Redaktion".

3 Forth News
Graeme Dunbar

VFX 3.6 für Windows (98, NT, 2000, ME, XP) - COMSOL-Newsletter - cfdos - kForth - F11-UK

4 AGM Report
Chris Jakeman <cjakeman@bigfoot.com>

Die Jahresversammlung der FIG UK fand am 25. Oktober 2003 in der Behausung von Douglas Neale statt. Dank geht an Doug und Mrs. Neale für ihre Gastfreundschaft. Die Zahl der Neuzugänge hält sich mit der Zahl der Austritte die Waage. Chris Jakeman arbeitet neuerdings als Lehrer in Computerdingen, mit geringerer Bezahlung und viel zu tun. Dank geht an Graeme Dunbar für die Weiterführung der Redaktionsarbeiten. Weitere TOPs: Buchverleih, F11UK, www.fig-uk.org. Nicht jeder möchte in comp.lang.forth "posten", ein Bulletin Board wäre empfehlenswert. Jeremy Fowell wird die Feierlichkeiten zum 25. Jubiläum im November 2004 in Birmingham organisieren. Eine Forthwrite-CD mit weiterem Material ist im Entstehen begriffen.

6 EuroForth 2003 - The Report
Howard Oakford <www.inventio.co.uk>

Die Tagung fand vom 17. bis 19. Oktober 2003 in Ross-on-Wye, UK, statt. 16 Teilnehmer aus 8 Ländern - 6 aus akademischen Institutionen, 10 aus Industrie und Wirtschaft. Aus unserer FG mit dabei: M.A. Ertl über "Superinstructions".

Der Rezensent: Ein Fachfremder, der sich nur mal so nebenbei, aus Spaß an der Freud', mit Forth beschäftigt, hat da wohl keine Chance? Und wie ist es mit der FG? Es ist nie so richtig geklärt worden: Wollen wir mit unseren Tagungen der euroForth nacheifern? Wollten wir nicht auch die Anfänger und Amateure stärker einbinden? Wie und wo? Schon wieder ist einer ausgetreten. Sie laufen uns davon!

10 A Virtual Nondeterministic Machine in Forth
James A. Boyd <JimBoyd@techemail.com>

Der Autor beschäftigt sich seit 1991 mit Forth als Hobby. Im Artikel wird das 8-Damen-Problem angesprochen. Ich (der Rezensent) übersetze einfach mal den Vorspann:

Eine nichtdeterministische Maschine ist eine faszinierende theoretische Konstruktion, die in der Lage ist, für jedes Element einer Lösung eines gegebenen Problems einen "richtigen" Wert zu finden, falls es zu dem Problem überhaupt eine Lösung gibt. Algorithmen für eine nichtdeterministische Maschine können beträchtlich schneller sein als deterministische Algorithmen. Der vorliegende Artikel beschreibt die Implementierung einer virtuellen nichtdeterministischen Maschine, ein Experiment in Forth, das, so ist zu hoffen, wesentliche Züge der theoretischen Maschine wiedergibt. Anstoß zu dieser Arbeit gaben das Buch über nichtdeterministische Algorithmen von Ellis Horowitz und Sartaj Sahni und ein Artikel über nichtdeterministische Steuer-Worte in Forth von L.L. Odette in Dr. Dobbs Journal aus dem Jahre 1983.

21 Library Notes
Graeme Dunbar

Furchtbares ist geschehen: In der School of Engineering an der Robert Gordon University wurde ein Gebäudekomplex renoviert, das gesamte Material des Buchverleihs der FIG UK musste umziehen, Graeme Dunbar war in Urlaub - und einiges ist verloren gegangen. Graeme befürchtet, für immer im Magen eines Reißwolfs. Unter anderem sind jetzt alle Unterlagen über ausgeliehene Literatur nicht mehr auffindbar. Graeme ruft die Mitglieder auf, ihm bei der Rekonstruktion der früheren Verhältnisse zu helfen.

Der Rezensent: "Zufällig" und kaum bemerkt, lagern bei uns in der FG die Unterlagen des Forthbüros im Keller des Rezensenten. Nicht auszudenken, wenn ... Verständlich, dass nicht jeder bereit ist, solche Verantwortlichkeiten auf sich zu nehmen. Aus derlei "Kleinarbeit" besteht aber das Vereinsleben.

22 Across the Big Teich
Henry Vinerts <Volvovid@aol.com>

Henrys Reports von September bis November 2003 über SVFIG-Aktivitäten in Originalfassung. Wir kennen sie in der Übersetzung von Thomas Beierlein. Diese Berichte werden in

der Forthwrite stets mit den folgenden einleitenden Zeilen veröffentlicht: This material was prepared for Vierte Dimension by Henry Vinerts, and printed by kind permission of Forth Gesellschaft (German FIG).

26 Letters

Graeme Dunbar: "Wir bedienen uns der unsterblichen Worte von Humphrey Lyttleton und dürfen sagen: Wir haben fast zwei Briefe in unserer Posttasche. Der vorliegende ist jedoch NICHT von einer Mrs. Trellis aus Nord-Wales." Es folgt ein Brief von Douglas Neale, der über 22 Bücher über Eingebettete Systeme berichtet, die kein einziges Wort über Forth enthalten.

27 Vierte Dimension 2/2003
Joe Anderson <jia@jus.abel.co.uk>

Joes Anstrengungen sind nicht hoch genug einzuschätzen. Diesmal eine kleine Ungenauigkeit, die jedoch im nächsten Heft über eine Korrekturmeldung leicht aufzufangen ist: Es ist das richtige VD-Heft besprochen worden, nämlich 2/2003, es wurde jedoch (auch auf dem Deckblatt) das Heft 4/2002 angegeben. Soweit zurück sind wir trotz der Umwälzungen in England (neue Redaktion, neuer Reviewer) nun doch nicht.

30 Deutsche Forth-Gesellschaft

Unsere Anzeige zur Anwerbung von Mitgliedern. Die Daten wurden bereinigt: Die DM-Angaben wurden in Euro umgewandelt und die Adresse des Forthbüros wurde aktualisiert.

31 What Languages Fix - Not!

Paul Grahams Bemerkungen im letzten Forthwrite-Heft darüber, dass neue Sprachen immer nur zu dem Zweck entwickelt werden, Fehler in bestehenden Sprachen zu beseitigen, war kein Witz - sagt Graeme Dunbar. Er ruft die Leser dazu auf, ein solches fehlerkorrigierendes Charakteristikum auch für Forth zu finden. Er gibt eine eigene Formulierung vor.

Einfacher Logarithmus

Rafael Deliano

Speziell für Dynamikkompressoren sind oft nur sehr grobe Näherungen nötig. Da die Datenworte dabei sehr breit sind, empfehlen sich Tabellen wegen des Speicherverbrauchs nicht.

scanbit

Die simpelste Näherung ist, sich bei einer positiven Zahl das höchste gesetzte Bit zu suchen (Bild 1). Die Funktion gibt es bei manchen 32-Bit RISC-Prozessoren im Befehlssatz, weil für Normalisierung von Floats nützlich. Dafür wird manchmal der Name *scanbit* verwendet. Entsprechend gibt es auch den Befehl *spanbit*, der die höchste gesetzte Null findet, für negative 2er-Komplementzahlen. Allerdings ist in Software wohl eine bitweise Invertierung des Datenwortes, gefolgt von *scanbit*, angemessener.

Bild1: scanbit

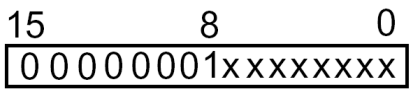
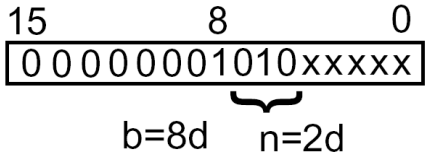


Bild 2: Umrechnung ln auf lb

$$lb(x) = 1{,}44269041 * ln(x)$$

Bild 3: bitlog 16



$$y = 8 (b - 1) + n$$

Die Werte von *scanbit* entsprechen der Funktion y= lb(x)+1, Wobei lb der binäre Logarithmus ist. Da Taschenrechner den manchmal nicht haben, ist die Umrechnungs-formel über ln für das Nachrechnen von Hand nützlich (Bild 2). Etwas problematisch ist der Datenverlust, wenn man 16 Bit auf 4 Bit staucht.

bitlog

Man kann das Verfahren natürlich verfeinern, indem man mehr Bit auswertet. Unter dem Namen *bitlog* ist eine solche Variante in [1] für 16 Bit Integer angegeben (Bild 3). Dabei werden die drei folgenden Bit hinter den führenden Bit zur Veredelung verwendet. Die Berechnung entspricht relativ gut y=8(lb(x)-1).
Bei Eingangswerten 0 .. 7 treten allerdings Probleme auf, so daß ein Patch nötig ist (Tabelle 2). Man spaltet diesen Bereich ab und berechnet das Ergebnis durch Shift*2. Man beachte den Unterschied der Kennlinie zum echten Logarithmus in diesem Bereich. (Bild 4): Letzterer läuft auf 1.

Die 16 Bit werden auf etwa 7 Bit gestaucht, der Maximalwert für y ist 119d. Das Verfahren läßt sich unschwer auf 32 und 64 Bit Datenworte skalieren (Tabelle 3). In Tabelle 4 sind Speicher und Laufzeit für Assemblerroutinen auf Mitsubishi 6502 mit 2,54 MHz Busfrequenz angegeben.

Tabelle 1:

y = scanbit (x)

x	y
0	0
1	0
2	1
4	2
8	3
16	4
32	5
...	
16384	15

Tabelle 2:

y = b116(x)

X x binär	b	n	y
8 ...1000	3	0	16
7 ...0111			14
6 ...0110			12
5 ...0101			10
4 ...0100			8
3 ...0011			6
2 ...0010			4
1 ...0001			2
0 ...0000			0

Tabelle 3: Skalierung von *bitlog* von 16 auf 32 und 64 Bit

```
y =  8(b-1)+n   b = 0 ... 15d   n = 3 Bit  x <  8d   y = x * 2   ymax =  119d
y = 16(b-2)+n   b = 0 ... 31d   n = 4 Bit  x < 16d   y = x * 2   ymax =  479d
y = 32(b-3)+n   b = 0 ... 63d   n = 5 Bit  x < 32d   y = x * 2   ymax = 1951d
```

Tabelle 4: *bitlog*

```
Speicher & Laufzeit
     Byte     µsec
BL16   60   25 – 100
BL32   90   50 – 150
BL64  120   50 – 250
```

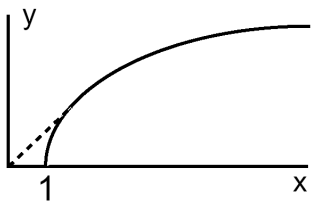


Bild 4: Kennlinien log & bitlog

Expander

Aus beiden Verfahren lassen sich auch Umkehrfunktionen bilden, die sich dann ähnlich Exponentialfunktionen verhalten. Für diese sind wegen der kurzen Wortlänge von x aber oft Tabellen möglich, wodurch man beliebige Kennlinien erzeugen kann.

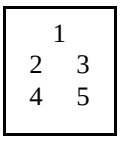
[1] Crenshaw; „MATH Toolkit for REAL-TIME Programming“ CMP-Books 2000

Listing 1: bitlog 16 auf 6502

```
<| \ BL16
:CODE BL16 \ ( UN1 — UC1 )
            \ Bitlog 16 bit
    BOT-  1+  ,X LDA, \ HB
3 $               BNE,
    BOT-      ,X LDA, \ LB
           08 #. CMP,
3 $               BCS,
        \ C=1 if 08 > data
              A. ASL,
    BOT-      ,X STA,
                 RTS,
3 $:        0F #. LDY,
1 $: BOT- 1+ ,X  LDA,
2 $              BMI,
    BOT-    ,X   ASL,
    BOT- 1+ ,X   ROL,
                 DEY,
1 $              BNE,
2 $:           DEY, \ n - 1
    N            STY,
 N ASL, N ASL, N ASL, \ *8
    BOT- 1+   ,X LDA,
A. LSR, A. LSR,
A. LSR, A. LSR,
       B% 111 #. AND,
                 CLC,
    N            ADC,
    BOT-      ,X STA,
           00 #. LDA,
    BOT-   1+ ,X STA,
                 RTS,
CODE; |>
```

Betreff: Hamilton und Euler
Von: behringe@mathematik.tu-muenchen.de

Hallo Michael,
aus der Fernsehwerbung kennen wir den Slogan "Wir bauen Ihrer Zukunft ein Zuhause". Fünf Punkte, die miteinander verbunden werden sollen. (Wenn man das "Haus" in Hundertwasserscher Manier oben breiter als unten baut und alle Punkte mit allen Punkten verbindet, wird ein Pentagramm daraus.)



Es ist leicht, einen Weg zu finden, der alle Punkte miteinander verknüpft, wobei, ohne alle möglichen Verbindungen ("Kanten" des Graphen) ausschöpfen zu wollen, jeder Punkt genau einmal berührt und zum Ausgangspunkt zurückgekehrt wird: Ein "geschlossener Hamiltonscher Weg" oder ein "Hamilton-Kreis".

Beispiel: 1 2 4 5 3 1 .

Die Fernsehwerbung verlangt, das Haus so zu bauen, das bestimmte, als wesentlich aufgefasste Kanten in einem einzigen Zug genau einmal durchlaufen werden, wobei die Punkte mehrfach berührt werden dürfen: "Das ist das Haus des Ni-ko-laus'."

Beispiel: 4 3 5 3 2 1 3 4 5 .

Das ist ein "Eulerscher Weg", aber ein "offener".
Zieht man noch Verbindungen von den unteren Eckpunkten zu den Giebeln ein, also vervollständigt man die Menge der Verbindungen zur Gesamtmenge sämtlicher Verbindungen im "Pentagramm", dann lässt sich auf diese Weise (ohne Absetzen, jede Kante genau einmal) auch zum Ausgangspunkt zurückkehren.

Beispiel: 4 2 5 3 2 1 3 4 5 1 4 .

Das ist ein "geschlossener Eulerscher Weg" oder ein "Euler-Kreis".
"In einem Zug" oder "ohne abzusetzen" kann man beispielsweise mit dem Bitterschen Turtle-Graphik-Lego-Roboter in die Praxis umsetzen, mit einem Apparat von der Art, wie er Bernd Paysan zu seiner Bemerkung in der VD 2/2002 verleitet und Martin zu seiner Gegenbemerkung in der VD 4/2002 veranlasst hat.

In Königsberg (Kaliningrad) am Pregel gibt es (gab es zumindest um 1740) sieben Brücken. Mit der Frage, ob es einen geschlossenen Eulerschen Weg über die Brücken gibt, fing die heutige Graphentheorie an.
Hamilton suchte um 1850 einen später nach ihm benannten Hamilton-Kreis im planifizierten Dodekaeder.

Und wenn mir noch einer sagen kann, wo ich den Befehlssatz des AMD64 her bekomme, bin ich für den Augenblick zufrieden. Bei AMD im Internet ist nichts zu machen. Mit Googeln kommt auch nichts ans Tageslicht. Ich meine nicht (nur) die auf die FPU-Register zurückgreifenden bekannten MMX-Befehle.

Herzlichen Gruß, *Fred*

Forth-Gruppen regional

Moers
Friederich Prinz
Tel.: (0 28 41) - 5 83 98 (p) (Q)
(Bitte den Anrufbeantworter nutzen!)
(Besucher: Bitte anmelden!)
Treffen: 2. und 4. Samstag im Monat
14:00 Uhr, **MALZ, Donaustraße 1**
47441 Moers

Mannheim
Thomas Prinz
Tel.: (0 62 71) - 28 30 (p)
Ewald Rieger
Tel.: (0 62 39) - 92 01 85 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München
Jens Wilke
Tel.: (0 89) - 8 97 68 90
Treffen: jeden 4. Mittwoch im Monat
Ristorante Pizzeria Gran Sasso
Ebenauer Str. 1
80637 München

Hamburg
Küstenforth
Klaus Schleisiek
Tel.: (**0 40) - 37 50 08 03 (g)**
kschleisiek@send.de
Treffen 1 Mal im Quartal
Ort und Zeit nach Vereinbarung
(bitte erfragen)

Gruppengründungen, Kontakte

Hier könnte Ihre Adresse oder Ihre Rufnummer stehen – wenn Sie eine Forthgruppe gründen wollen.

µP-Controller Verleih

Thomas Prinz
Tel.: (**0 62 71) - 28 30** (p)
micro@forth-ev.de

Forth-Hilfe für Ratsuchende

Jörg Plewe
Tel.: (**02 08) - 49 70 68** (p)

Jörg Staben
Tel.: (**0 21 03) - 24 06 09** (p)

Karl Schroer
Tel.: (**0 28 45) - 2 89 51** (p)

Spezielle Fachgebiete

FORTHchips
(FRP 1600, RTX, Novix)
Klaus Schleisiek-Kern
Tel.: (**0 40) - 37 50 08 03** (g)

F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020, TDS9092), Fuzzy
Arndt Klingelnberg, Consultants
akg@aachen.kbbs.org
Tel.: **(00 32)(0 87) - 63 09 89** (pgQ)
Fax: **- 63 09 88**

KI, Object Oriented Forth, Sicherheitskritische Systeme
Ulrich Hoffmann
Tel.: (**0 43 51) - 71 22 17** (p)
Fax: **- 71 22 16**

Forth-Vertrieb
vlksFORTH
ultraFORTH
RTX / FG / Super8
KK-FORTH
Ingenieurbüro **Klaus Kohl**
Tel.: (**0 82 33) - 3 05 24** (p)
Fax :(**0 82 33) - 99 71**
mailorder@forth-ev.de

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 Arbeitszeiten
g = geschäftlich

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

20 Jahre Forthgesellschaft

Bernd Paysan

Haben Sie sich schon zum

Drachenfest

angemeldet?

Hallo Forth-Interessierte,

in diesem Jahr feiert die Forth-Gesellschaft ihr 20 jähriges Bestehen. Die Forth-Tagung 20FG04 findet in der Zeit vom 16.-18. April 2004 in Burgstaaken auf Fehmarn statt. Am Samstagabend feiern wir mit einem Festmahl und mit musikalischer Darbietung.

Das Tagungsprogramm werden wir gewohnt locker gestalten, um Freiräume für Workshops und Diskussionen zu schaffen. Vortragende sollten bis spätestens 15.03.2004 ihre Anmeldung inklusive Abstract einsenden, damit wir den Tagungsablauf planen können.

Gesucht werden insbesondere Erfahrensberichte über Anwendungen, Projekte, Implementierungen, Systeme und Entwicklungstechniken rund um Forth und aus gegebenem Anlaß speziell auch historische Darstellungen der Forth-Geschichte.

Falls die Aufnahme eines Artikels zum Vortrag im Tagungsband gewünscht wird, muß dieser bis zum 15.03.2004 druckfertig vorliegen (DIN A4, 2cm Rand an allen Seiten, Fußzeilen mit Name und Thema, Seitennumerierung).

Weitere Informationen und Anmeldeunterlagen finden sich unter

http://www.forth-ev.de/tagung/einladung2004.html

Viele Grüße,
Ulrich Hoffmann

--

Dr. Ulrich Hoffmann
Sehestedter Strasse 26
D-24340 Eckernfoerde, Germany

Ulrich.E.Hoffmann@gmx.de
uho@xlerb.de
Tel. +49 4351 712-217, Fax -218