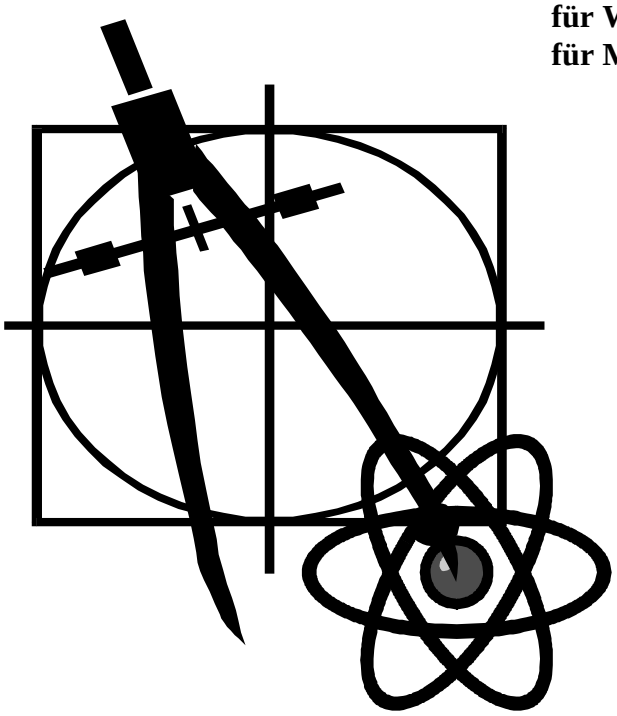
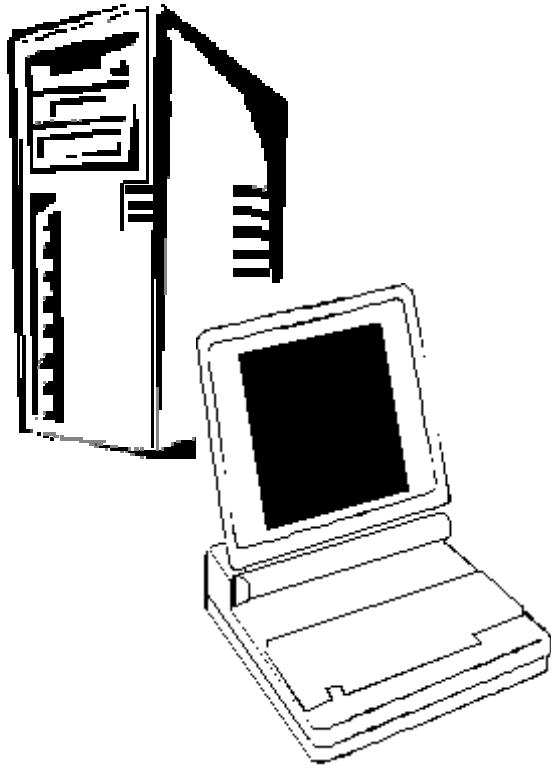
VIERTE DIMENSION

3 / 2003

Das FORTH-Magazin



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



In dieser Ausgabe:

Leserbriefe und Rezensionen

Leser schreiben, was sie interessiert

Holon lebt

Neues zum „Schweizer Forth“

Compiler / Interpreter

Eine leidenschaftliche Unterweisung

Lebendiges Forth

Neues aus den USA

Vierte Dimension

Fortsetzung der Titelliste

Forth auf dem LINUX-Tag

Berichte und Leserbriefe

pOOP in Forth

OOP auf der Basis von Prelude

MicroCore Philosophie

Freigabe der Lizenz

Pflock-Solitaire und TRICEPS

Der spielende Roboter

19. Jahrgang - **Ausg. 3/2003** - 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

Impressum 4

Editorial 4

Leserbriefe 5

Holon lebt, ***Wolf Weijgaard*** 6
Versionsverwaltung mit HolonX

Begriffsklärung, Compiler / Interpreter, ***Ewald Pfau*** 7
Eine leidenschaftliche Unterweisung

Gehaltvolles, ***Fred Behringer*** 12
Rezensionen unserer Schwesterjournale

Lebendiges Forth, ***Henry Vinerts*** 13, 22
Lebenszeichen aus den USA

Vierte Dimension, ***Fred Behringer*** 14
Fortsetzung der Titelliste

Zum LINUX-Tag, ***Carsten Strotmann*** 16
Eine Anregung für die nächsten Jahre

pOOP in Forth, ***Manfred Mahlow*** 17
Objektorientierung auf der Basis von *prelude*

MicroCore Philosophie, ***Klaus Schleisiek*** 21
Die Freigabe der MicroCore Technologie

Was ist Pflock-Solitaire?, ***Ewald Rieger*** 23
Grundlagen zum Spiel des TRICEPS

TRICEPS spielt Solitaire, ***Ewald Rieger*** 25
Implementierung von Solitaire für TRICEPS

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:

- Fletcher-Prüfsummen, einfacher als CRC
- P-Code Systeme
- ein neuer/alter Stackprozessor von Echelon
- Ulrich Pauls Target Compiler

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,

Gerade einmal 32 Seiten ist die vorliegende Vierte Dimension dünn. War jemals eine Ausgabe unseres Magazins magerer?

Die Ausgabe 04 2003 kann noch erheblich schmaler werden. Einige wenige Zusagen für Beiträge liegen in der Redaktion vor. Zusagen, wohlgemerkt, leider keine Artikel.

Was ist denn nur los mit den deutschen Forthern? Die Mitgliederzahlen der Forth-Gesellschaft steigen erstmals seit vielen Jahren wieder. Die Kontakte in alle Welt, in der sich so viel Interessantes und Bemerkenswertes tut, sind so gut wie nie zuvor. In den USA regt sich ein tot geglaubtes Pflänzchen wieder. Nur die FG scheint in einen tiefen Schlaf gefallen zu sein.

Ich hoffe sehr, daß nur der phantastische Sommer mit seinen Hitzerekorden die Lust am Schreiben ein wenig gedämpft hat, und daß schon in den nächsten Tagen nach der Veröffentlichung dieser Ausgabe neue Beiträge in großer Anzahl hier eingehen.

MinForth wäre sicher etwas, worüber sich zu berichten lohnt. Die Adressen im Internet, die in dieser Ausgabe zum Beispiel von Henry Vinerts genannt werden, geben mehr als nur einen spannenden Beitrag her. Und haben Sie schon einmal die „Forth-Schatzinsel“ bei *www.stejskal.de/web/computer/forth/index.html* besucht?

Schauen Sie, was die Welt uns bietet – und berichten Sie darüber. Bitte.

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: Brief eines Lesers
Datum: Thu, 29 May 2003
Von: Fred Behringer

Lieber Fritz,
die VD ist gut, der Inhalt beeindruckt mich, die Organisation klappt. Was wollen wir mehr? Es tat sich allerhand, und trotzdem sind wir froh, dass eigentlich alles so geblieben ist, wie es schon immer war.
Interessant ist die eine Zuschrift von Ulrich Paul. Lübke-Englisch als Art "bytecode", als Vor- oder Zwischenstufe - als Vorstufe zu einer kontextsensitiven Übersetzungsmaschine. Vielleicht ruft man sich hierzu auch mein "Umlautproblem" aus der VD 3/2000, S.28, ins Gedächtnis. Eine Vorstufe zur Vorstufe, aber trotz der niedrig angesetzten Messlatte schon eine, die nicht ganz ohne Kontextberücksichtigung auskommt. Herbert Fink war seinerzeit fleißig (VD 4/2000). In VBA-Skript - nicht in Forth. Nun, vielleicht hat Ulrich Paul mehr Glück. Ich bin jedenfalls interessiert. He is heavy on wire. I draw me the jacket on and will me in Denglish try. Will this the language be which we in fifty years speak will? Der arme Henry Vinerts! Wie will der das seinen SVFIG-Kollegen erklären? Vor einiger Zeit hat ihm die "volle Verarsche" Rätsel aufgegeben. Und jetzt auch noch das! Forth for Fun! Packen wir's an!
Forth ist vielseitig. Die Interessen der Forthler sind breit gefächert. Und die "Vierte Dimension" spielt bei der Vermittlung eine zentrale Rolle.

Herzlichen Gruß

Fred

Betreff: Forth auf dem Linux-Tag
Von: Bernd Paysan <bernd.paysan@gmx.de>

So, die Forth-Gesellschaft war auf dem Linux-Tag.

Hier mal ein Kurzbericht:
Unser Blickfang war der Triceps, der unermüdlich immer wieder das gleiche Pflock-Solitaire-Spiel gespielt hat (und nebendran ein Laptop die Emulation mit OpenGL-Visualisierung - der hat sich hin und wieder ein neues Spiel ausgerechnet). Diese Strategie ist definitiv aufgegangen, der Stand war ständig belagert, und die "langhaarigen Linux-Lemminge" (Heise.de-Flamewar-Jargon, die Anzahl langhaariger bleicher Hacker ist auf dem Linuxtag aber schon recht hoch ;-) haben uns auch Löcher in den Bauch gefragt, wie "Gewinnt der [Roboter] da jetzt auch?", "Forth - muß man das kennen?", "Was sind die Stärken von Forth", "Wo wird Forth eingesetzt" und natürlich "Wie sieht da der Quellcode aus?".

Das Wochenend-Publikum hat nicht so viel und vor allem nicht so tiefgründig gefragt, dafür uns die Flyer aus den Händen gerissen.
Das b16-Board haben wir auch gezeigt, und dank Blickfang haben die Leute es auch entdeckt (obwohl es so klein ist). Und natürlich haben wir kostenlose Verbesserungsvorschläge bekommen ;-):

* Eigentlich sollte ja der b16 den Roboter steuern. Mit der nächsten Board-Version geht das auch. Und die Zuschauer fänden es spannender, wenn der Roboter nicht nur ein vorprogrammiertes Spiel spielt...

* Die 33. Kugel sieht immer etwas verloren aus. Mein Vorschlag: Die Kugeln im Kreis anordnen (32 Kugeln), das ist dann sogar noch leichter zu berechnen - und Friedls Blech-Stanze kriegt das sicher auch hin.

Bernd Paysan

Photos: Thomas Prinz

Holonforth® HOLON lebt

Betreff: Forth, HolonX und Change Control
Von: Wolf Wejgaard <www@tiredofspam.com>

Mit einem Gruss aus der Holon-Ecke würde ich heute gerne ein Thema aufgreifen, das eigentlich die Forthwelt interessieren müsste, obwohl es kaum besprochen wird, soweit ich mich erinnere. Ich meine das Verwalten von Aenderungen an einem Programm.
Auch Forth-Programmierer ändern ja ab und zu ihre Programme – vielleicht sogar öfters als andere, weil es sich in Forth so leicht und freudig ändern lässt.

Merkwürdigerweise hatte ich mit der "change control" immer mehr Schwierigkeiten als mit der reinen Software. Im Programm selbst finde ich zu jeder Aufgabe meist auch eine befriedigende Lösung, wenn ich lange genug hinschaue. Dagegen komme ich mit der Verwaltung der Aenderungen leicht ins Schleudern und vergesse mal ein Wort zu testen. Geht das anderen auch so, oder bin ich einfach zu wenig diszipliniert?

Wie machen das die Spezialisten?

Solange ich die Aenderungen separat vom Programm verwalte, muss ich zwischen zwei Domänen hin und her wechseln. Das lenkt immer wieder vom Programm ab und ich verdränge wohl ab und zu das Nachtragen einer Aenderung in der "Buchhaltung". Ideal fände ich eine Methode, die beides, Quelltext und Aenderungskontrolle, vereint. In dem Sinne, dass bei einer Aenderung an einem Wort automatisch ein mit dem Wort verbundener Zustand sichtbar geändert wird, um es erst einmal vage auszudrücken.
Nun, ich habe seit langem den Verdacht, dass Forth dank seiner Datenbanknatur auch punkto „Change Control“ elegante Möglichkeiten bietet, und möchte hier eine Lösung vorstellen, die ich jetzt in HolonX realisiert habe.
Habe ich HolonX hier schon vorgestellt? Kurz gesagt, es ist eine universelle Version von Holon, die nur den Quelltext verwaltet, in der bekannten Struktur. Beim "Laden" des Programms schreibt HolonX den Quelltext zusammenhängend in ein ASCII Textfile, das dann einem beliebigen Interpreter/ Compiler übergeben werden kann.
Das kann ein anderes Forth System sein, aber auch eine fremde Sprache. HolonX ist im Web bei ***holonforth.com*** verfügbar, als Freeware. - Uebrigens sind seit April 2003 auch die anderen Holonversionen, die in der Website zur Verfügung gestellt werden, frei verwendbar.

Zur Aenderungskontrolle. Ich denke an zwei Aspekte: Version und Validierung. Nehmen wir an, eine Version sei fertig und ausgeliefert und nach einiger Zeit wird die nächste Version fällig. Es gibt Bugs zu erschlagen, Korrekturen und neue Funktionen zu programmieren. In Forth bedeutet das, dass viele Worte berührt werden, die zum Schluss der Uebung alle getestet und für gut befunden (validiert) sein sollten. Ich möchte also sehen, welche Worte zur neuen Version gehören und in welchem Zustand jedes dieser Worte ist.

Wie bereits angetönt, bietet Forth in seiner Datenbank - auch als Dictionary bekannt - die Möglichkeit, neue Felder einzusetzen und damit den Zustand eines Wortes zu markieren. Ein Feld könnte eine Versionsnummer aufnehmen, ein anderes eine Zustandsvariable. Es braucht dazu ein neu generiertes Forthsystem, ok, aber das ist ja machbar.

Im klassischen Forth (Quelltext getrennt vom System verwaltet) könnte man im Quelltext zu jedem Wort entsprechende neue Compilerworte einfügen und damit beim Laden die Felder füllen.

Beispiel: Version: xxx Zustand: y.

Mit ein paar einfachen Routinen könnte man sich dann nach dem Laden, wenn das Dictionary existiert, Listen der neuen Wörter und ihres Zustandes produzieren lassen.

Holon verwaltet den Quelltext bereits mit dem Dictionary zusammen und kann daher das Ganze integrieren. Die Lösung in HolonX sieht nun so aus: Mit dem Festlegen einer neuen Versionsnummer im Systemmenu wird auch der Zeitpunkt festgehalten (Datum und Tageszeit). Die Worte speichern ebenfalls die Zeit der letzten Aenderung des Zustandes. Der Vergleich der Zeiten zeigt, welche Worte zur neuen Version gehören. Die Zeit (Datum) wird im Browser neben dem Wort gezeigt in Farben, die dem Zustand des Wortes entsprechen:

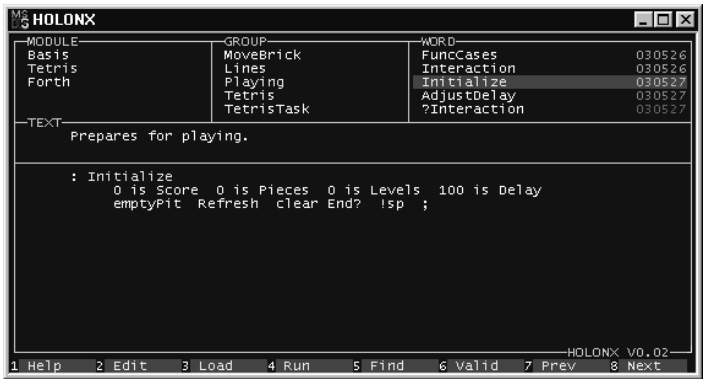
rot = editiert,
blau = geladen,
grün = validiert,
grau = validiert

in früherer Version. Damit lässt sich der Zustand des Programmes schnell erkennen.

Ich unterscheide zwischen "editiert" und "geladen", weil Holon das laufende Zielprogramm wortweise ändern kann. Dann ist es gut zu wissen, ob das geänderte Wort auch geladen und im Zielsystem wirksam ist, und ich also wirklich die neue Variante teste.
Mit diesem Zusatz zu HolonX macht das Aendern eines Programmes erstmals ungetrübtes Vergnügen, finde ich. Sobald ich nur noch grüne oder graue Timestamps sehe, ist eine neue Version beendet.

Das System steht, wie gesagt, zum Probieren und auch zum Verwenden frei zur Verfügung bei holonforth com. Ich betrachte es immer noch als ein Experiment und nehme Anregungen und Kommentare gerne auf.

Wolf Wejgaard



Begriffsklärung, Compiler / Interpreter

Betreff: Artikel der *englischen* Lehnwörter
Von: Ewald Pfau <ehp@ear.co.at>

Hintergrund: Beim Lesen von de\comp\lang\forth ist mir vor einigen Wochen ein sehr leidenschaftlicher Beitrag aufgefallen, der versehentlich in „unserem“ Forum gelandet war. Diesen Beitrag wollte ich den Lesern der VD zugänglich machen – wozu der Autor auf eine entsprechende Anfrage postwendend sein Einverständnis erklärt hat.

fep

Josef 'Jupp' Schugt wrote:

> 'Kern eines jeden LISP-Systems ist ein Interpretierer (auch
> Interpreter genannt), der eingegebene Ausdruecke evaluiert
> (d.h. auswertet) und die Resultatwerte ausgibt.'
>
> Das mit dem Compilieren ist demnach neumodischer
> Schnickschnack...

Warum sich so leicht verwirren lassen, wenn man es hier sprachlich aufrollen kann? Das muss so ein Zwang zur Ingroup-Kohaerenz sein, dass das mit dem Interpretieren / Kompilieren nicht viel unmittelbarer beim Schopf gepackt wird.

Solche Ingroup-Kohaerenz verlangte es wohl, dass 'das Programmieren' im hehren Sinne zu etwas zunutze sei, so dass im Resultat Programme vorlaegen, Datenhaufen besonderer Art jedenfalls, die nicht so einfach einsichtig sein duerften. Und in diesem Sinn ist ein resultierendes Programm ein Industrieprodukt, das sich in Plastik einwickeln und dann in einem Kaufhaus praesentieren laesst. Dazu muss man es kompilieren.

Seit sich dieser Kontext etabliert hat, 'Programm' plus dunstiger Datenhaufen plus 'kompilieren', steht der Vorgang des Kompilierens offenbar unter besonderem Artenschutz. Erst mit der Debatte um die Open Source wird der teilweise Unsinn (logisch gesehen), der da verbarrikadiert ist, neu aufgerollt und in ein ertraeglicheres Licht gestellt. Dass von solchen proprietaeren Datenhaufen der Ausgang von Wahlen abhaengen soll, dass dieserart die ausprogrammierten Gedanken tabu seien, dass alleine eine Maschine zur Ueberpruefung anstehe, niemals jedoch der Gedanke, der darin verbaut wurde, dass, auf diesen Gedanken zu schliessen, mehr und mehr zudem kriminalisiert werde – wuerde in jeder anderen Industriesparte wohl eiligst als horrender Unsinn klargestellt.

Vor allem ist mit der Open Source-Bestrebung wieder klarer, dass solche ominoesen Datenhaufen nur zufaelligerweise solche sind und eigentlich sollte es um den Austausch von Gedanken gehen. Wie diese dann in die Maschine kommen, ist eher zweitrangig (wenn denn die Laufzeitumgebung, politisch gesehen, gesichert ist, was mit Linux und den BSDs und der GNU-Umgebung der Fall ist und hoffentlich auch so bleibt). Ein Compiler ist ein wenig etwas anderes, auch wenn er sich technisch nicht geaendert hat.

*****

Mit den Kommandozeilen ist immer ein Interpreter im Spiel, aber der ist ein Stueck ins Hintertreffen geraten, soweit die Anwender auf dem Schirm wohl lieber Kino schauen als (scrollenden) Fliesstext lesen. Da sind noch zwei Umgebungen, wo ein Interpreter zugleich die Programmiersprachenumgebung bereitstellt und den Anschluss ans Betriebssystem (mit Zugriff auf die angeschlossenen Ressourcen). Der Sinclair QL (ich wiederhole mich ;)) hatte ein recht maechtiges BASIC zur Grundlage, und der Austausch zum Gebrauch der Maschine trug Blueten, die heute wenig mehr vorstellbar sind, da ging es weniger darum, dass die Maschine funktioniert (also dass der Auspuff richtig angeschraubt ist) und dazu oberaffengeil sei usw., sondern logische Raetsel wurden ausgetauscht, Spiele im Austausch ueber Zeitschriften entwickelt und solche Dinge, in etwa Automechanik fuer Fortgeschrittene nur eben Tastatur statt Schraubenschluessel. Der Begriff „Fortschritt“ als staendig treibendes Vehikel wurde, im Vergleich zu 15 Jahre spaeter, recht wenig gebraucht bzw. viel harmloser konnotiert.
So ein Maschinen-Interpreter ist etwas Feines, wenn es genuegend solche gibt, die ein wenig Phantasie darauf verwenden. Gleichzeitig mit der Dominanz von MS und den PCs sind viele solche Spielwiesen wohl eingetrocknet (in Newsgroups findet sich zwar das Potential, aber dass es sich faktisch aeussert, kann man so nicht erwarten, zudem hat der Zahn der Zeit wohl jene Kombination von Geduld und unbekuemmerter, in logischer Abstraktion begruendeter Verspieltheit ziemlich abgenagt).

Dazu gibt es eine spezielle Sichtweise mit der Programmierumgebung von Forth. 'Kompilieren' ist hier nur ein voruebergehender Zustand des Interpreters, um quasi ein benanntes Makro abzulegen (das in Forth 'Word' heisst). Damit wird der Vorrat an Moeglichkeiten, die der Interpreter bietet, erweitert. Gleichzeitig hat aber der Compiler keine Vorrechte, sondern ist ebenso erweiterbar. (Und weil den meisten Programm-Programmierern dies wohl aeusserst suspekt ist und meist zudem einem Management ein Chamaeleon ebenso, ist die Forth-Umgebung zwar bekannt, aber kaum jemand kennt sie).

Fuer sprachliche Betrachtungen ist das nun insoweit interessant, als 'Kompilieren' ('auf einen Haufen werfen') nun sehr transparenterweise Abstraktion und Verzoegerung darstellt. Und in Gang geworfen wird eine Machine letztlich durch einen Interpreter-Aufruf. Die Denkweise in Forth fordert, sich das transparent vorzustellen: wenn es auf die gemuetliche Tour zu langsam ist, dann wird die Anzahl der ad hoc kompilierten Worte vergroessert; wenn es dann immer noch zu langsam ist, dann werden einzelne Worte in Forth-Assembler direkt auf den Prozessor geschneidert.

Ob dann ein Saegewerk in Betrieb ist, dieweil der Programmablauf via Quelltext von einer Floppy gelesen wird (was dem Vernehmen nach gut und zuverlaessig funktioniert) oder wie immer sonst das geht: Mit dem geschilderten Hintergrund entlarvt sich die Frage, ob es denn kompiliert sei oder nicht oder P-Code oder was immer, als reichlich ideologisch unterminiert.

Eine leidenschaftliche Unterweisung

Diese Auseinandersetzung muss auf etwas Anderes abgezielt haben, vielleicht dem, dass per hegemonieller Gravitation (die ja wohl bei der Rechnerei viel wichtiger ist als alle Logik zusammen) die Tatsache staerker ins Zentrum gerueckt ist, dass das Resultat einer Arbeit in Form eines Programm vorliegen muss. Ein Programm ist ein Programm, und das muss man kompilieren, sonst ist es kein Programm. Oder so irgendwie...

Der Vorgang der Instruktion einer Maschine wird hiermit zur Gaenze in die Planung verlagert. Es gibt dann nur zwei Zeitpunkte der Betrachtung: der Zeitpunkt der Vorwegnahme, wenn ein Programm geschrieben und kompiliert wird und der Zeitpunkt, dass es ausgefuehrt wird.

Bei Interpretern geht das nicht. Schon gar nicht, wenn ein Interpreter waehrend er interpretiert auch noch am Compiler herumstricken kann, da wird der 'Zeitpunkt, dass ausgefuehrt wird' sehr diffus (man kann sich sich, so man will, einen Comnpiler fuer Klassen und Methoden stricken und dann objekt-orientiert weiterprogrammieren oder einen Basic-Interpreter kompilieren und damit dann den folgenden Text im Interpreter als BASIC uebersetzen - nur um eine Idee zu geben; die logische Crux ist, dass der Compiler per Uebereinkunft trivial ist).

Vielleicht gibt es deshalb so leicht gereizte Reaktionen (soweit mir in diesem Zusammenhang ebenfalls gelaeufig und wie hier auch gleich wieder zu finden, wenn so etwas aufkommt).

Martin Gerdes wrote:

> Schriebs und verwirrte in der ihm eigenen Weise.

Vielleicht haette Claude Levy-Strauss ueber Programmierer schreiben sollen und nicht ueber ferne Voelker? Vielleicht sublimiere ich nur, dass mir das abgeht? Oder hast Du nur einen beigestellten Schaukelstuhl vermisst, wg. gemuetlicher?

> : achso ." Das erklaert manches!" ;

Na ei-der-daus aber auch! 8]

> Warum nicht? Du muesstest das Follow-Up halt nur setzen.
Waere besser. Es waren aber zwei Nennungen in der 'Newsgroups:'-Zeile.

Josef 'Jupp' Schugt wrote:

> Offenbar gibt es auch noch andere Auslegungen der beiden
> Begriffe. Es waere schoen, diese einmal zu sammeln...

Ja denkst Du ich schreib den ganzen Sermon von gestern, Antwort an Dich, grad noch einmal? Ja: dort war von Interpreter und Compiler gerade etwa so die Rede, wie Dein obiger Satz geschnitten ist. Hmm.

(erst noch einen Lehnstuhl fuer MG, bevor noch weitere Absaetze folgen... so ... ;) ...)

Die Bedeutung dieser Begriffe ist eigentlich stabil. Was sich wandelt, das ist nicht die normalsprachliche Bedeutung, viel mehr an welcher Stelle im Umgang mit einer Maschine die entsprechenden logischen Instanzen umgesetzt sind, und wie das geschieht, in welcher Form das zu erwarten ist, welche Konsequenzen fuer den Umgang mit der Maschine das hat.

Die Frage zur Auslegung ist dann die nach der Tragweite. Dazu muss man sich auf Details der Bedeutung einlassen, sie fuer unterschiedliche Betriebsbedingungen zu deuten wissen. Bei Compiler & Interpreter duerfte das nicht so schwer sein. Vielleicht ein springender Punkt ist hier der rekursive Moment, wenn eine Sache sich selbst aus dem Sumpf ziehen soll.
Dort kulminiert auch die Terminologie und muss praeziser sein als sonst. Das aeltere Baendchen 'lex & yacc' behandelt so etwas (ueber Yet Another Compiler Compiler).

Wenn Du also nach Auslegungen der Begriffe im weiteren Sinn suchst, dann, schaetze ich, wirst Du im Laufe der Zeit vor allem so einem seltsamen Zirkus begegnen, zu der Zeit, als diejenigen von den Wortbedeutungsfabriken sich allzu oft in Imperativen ergangen haben und ebenso gar oft den Satzbau des vollendeten Futurs strapazierten. Merkmal mithin fuer hochgradige Euphemismusschaedigung in bedenklichem Stadium.

Da ist dann nicht 'Auslegung' wichtig, weil ja technische Details mit der Handbewegung des Imperators in Richtung des Dienstpersonals abgehandelt werden. Es waren die Euphemismusproduzenten doch eigentlich selbst die Bevormundeten, wenn es schwierig ist, Ueberblick zu bewahren, waehrend sich die Gewichtungen staendig verschieben koennen, und da muessen wohl die serioesen Betrachtungen ausgesiebt werden aus dem staendigen vom Virus versetzten Uberschwang. Normalsprachlich. Schaumschlaegereien, Weltenszenarios, allesamt auch kunstvoll verschluesselte Allmachtsgelueste. Das dauert ja auch noch an.

Ein Skandal wie derjenige, dass durch eine Hintertuer die Errungenschaft der Demokratie ploetzlich verschwinden koennte, so ganz aus Versehen, ist in dem Zusammenhang wohl leider immer noch allzu real und tatsaechlich vorstellbar. Der Wechsel in den euphemistischen Modus passiert unbedacht, wenn auch von serioesen Medien reichlich blind Modernitaet eingefordert wird ohne ueberhaupt zu pruefen, wovon denn im Detail die Rede sei.

Es ist also eine Phase vorher interessant, wenn man sich fuer eine normalsprachliche Auslegung interessiert. Bis Mitte der 1970er und ein gutes Stueck danach waren Prozessorzeit und Speicher recht knapp, mit Anweisungen moeglichst nah ausgearbeitet fuer das logische Geruest einer CPU.

*****

Nun kommt aber das Werkzeug ins Spiel, wenn von Compiler die Rede ist, die Konvention, wie die Anweisungen geschrieben werden sollen. Und fuer eine ganze Reihe solcher Konventionen ist dies dann der Arbeitshorizont, dass Programme im

Quelltext geschrieben und dann mit dem Compiler fuer das logische Geruest einer CPU uebersetzt werden. Gleichzeitig koexistierten zwei andere Auffassungen, die sich beide einen Schritt von der strikten Bindung an die CPU loesten, im einen Fall durch Einfuehrung einer Zwischenschicht, in die uebersetzt wird (das war frueher P-Code, heute ist es der Code-Strom fuer virtuelle Maschinen), und eine weitere Auffassung, wo es gleich der lesbare Quelltext selbst ist, womit die Maschine zur Laufzeit eines Programms gefuettert wird.

Im Detail betrachtet, wird immer interpretiert, ausser der Strom von Anweisungen, wenn dieser fuer die CPU-Architektur vorliegt, erst in letzter Instanz wird nicht mehr interpretiert, sondern dekodiert, Anweisungen, Adressen und Werte aus Zahlenmustern, die aus dem Speicher gelesen werden. Wenn der Strom von Anweisungen in so eine Form gebracht wird, die eine logische Abstraktion auf unterster Ebene einfuehrt, so wird bereits dieser nun abstraktere Strom von Anweisungen auf Maschinenebene interpretiert. Das macht Sinn fuer die normalsprachliche Verwendung. Wenn dieses Spiel einen Schritt weiter getrieben wird, so besteht der Strom von Anweisungen aus dem Quelltext oder einer Zwischenstufe. Die Umsetzung geschieht zur Laufzeit, das kostet extra Rechenzeit.

(Und ein Wort zu Forth hier, weil damit das Spiel trivial und transparent vor sich geht, es somit ein nettes Modell abgibt: Der Quelltext braucht keine Vorbehandlung, sondern kann gleich an die CPU verfuettert werden, dazu muessen aber Rechenaufgaben in umgekehrt polnischer Notation geschrieben werden, dann geht das; kompilieren heisst hier, ein Stueck des Quelltextes in einer Zwischenstufe abzuspeichern und alles was bislang in der Zwischenstufe steht, wird als 'Woerterbuch' aufgefasst, das von der CPU leichter abzuarbeiten ist, und als 'Kompilieren' wird hernach jede Aktion verstanden, mit der was auch immer „an das Woerterbuch angefuegt“ wird.)

Um das Wort „Compiler“ rankt sich wohl vor allem deshalb ein Zirkus, der sich verselbstaendigt, soweit einerseits auf Maschinenseite der generierte Code staendig besser an die CPU angepasst wird, gleichzeitig die Idee, was eine Programmiersprache ausmacht, irgendwann begann, in illusorische Sphaeren abzuheben, damit aber 'Der Compiler' zu einer Instanz wurde, um solche Gegenlaeufigkeit zu fassen.

Es haben sich wohl die Maschinenressourcen soweit verschoben, dass nicht mehr mit jedem Prozessortaktschlag gegeizt werden muss, womit eine groessere Auswahl an Interpretersprachen sich breitgemacht hat (Perl, Python, PHP, SQL, Java usw: das sind alles Interpreter-Umgebungen, zum Teil mit zusaetzlich beigefuegter virtueller Maschine fuer Zwischencode). In der Konsequenz koennte sich die Bedeutung der Begriffe leicht verschieben (z.B. dass naheliegend sein koennte, 'Interpreter' weniger im Gegensatz zu 'Compiler' zu verstehen).

Zum Weiteren macht sich die Sitte breit, dass mehr und mehr von einer Seite einer Maschine ein Fliesstext generiert wird, der in eine andere Seite eingefuettert wird. Wenn man sich amuesieren will, kann man das auch selbst schreiben, allerdings sind das ziemliche Datenmassen geworden.
Dazu gehoeren vor allem SQL, HTML, PostScript.

*****

Wo der Compiler zusehr als abgehobenes Werkzeug verstanden wird, von Mensch zu Maschine, da runzelt sich mir leicht die Stirn, insoweit, als das Vorhandensein eines Compilers das Verstaendnis dessen, was einen Rechner ausmacht, in eine Richtung gedraengt hat, fuer das es sonst keine Notwendigkeit gibt, das ist vor allem die Fixierung auf Programme als eine Art undurchschaubare Geheimnistraeger. Damit konnte sich ein Begriff von Software auf eine Art industriell breitmachen, wie es sonst nicht so leicht einsichtig waere. Mit der Open Source-Debatte geht es hiermit doch eher wieder zurueck, mehr wieder in Sphaeren der Motivation durch Vernunft statt des Verdunkelns und Versteckens mit obskur scheinendem Datensalat, wenn man denn das Arbeitsgeraet zusehr in den Mittelpunkt gerueckt sieht, wozu die Verbreitungsbedingungen von Programmen zusaetzlich animierten, als quasi unlesbarer Datensalat, der nur fuer eine Maschine bestimmt sei. Der Compiler verliert diese besondere Stellung in dem Moment, wo die Verbreitung von Programmen ueber Quelltexte geschieht, wonach einsichtig ist, dass es eigentlich festgeschriebene Gedanken sind, die die Maschine zum Laufen bringen (und der Compiler gewinnt eine andere bevorzugte Stellung, indem die ausgetauschten Quelltexte lauffaehig gemacht werden muessen, womit das Werkzeug auf jeder Maschine gebraucht wird, auch wenn eigentlich niemand programmiert).

Diese real entfremdete Stellung von lauffaehigen Programmen, der nun wieder entgegengearbeitet wird, das ist der Stoff, aus dem Sabotage und Misstrauen sind und Bespitzelung als Gegenmaßnahme. Das ist eine direkte Verbindung von einer bestimmten Auffassung, gewohnheitsmaeßig eingebuergert, eines Werkzeugs. Eine Zeitlang habe ich mich ueber eine weitere Erscheinungsform noch geaergert: wenn man einen Programmierer fragte, was 2 plus 2 sei, dass die Antwort in der Formulierung besteht, die er seinem Compiler verfuettern wuerde. Man bekommt keine Auskunft. Man bekommt die Entfremdung, vorgefuehrt als Normalitaet.

Dem Interpreter kommt solcherlei kaum zu. Der ist in dieser Verbreitung zeitlich juenger (obwohl der Vorgang in dem Zusammenhang dennoch aelter ist), aber das birgt noch herbe Ueberraschungen, als sich abzeichnet, dass fuer Datenmassen die allzu freizuegig herumliegen, wohl mit aller Tuecke und Intrige das roemische Herrschaftsprinzip dazwischengekeilt werden soll, um Zugehoerigkeiten zwangsweise und kuenstlich herstellen zu koennen. Diese Suppe ist wohl noch nicht gegessen. Die Schiebereien finden aber gerade derzeit statt.

Da besteht eine direkte Verbindung mit dem Werkzeug 'Interpreter' (z.B. um eine Klangdatei abzuspielen oder Webseiten besuchen). Aus der Zeit der hochtrabenden Euphemismen ist noch der Unsinn verankert, dass 'natuerlich' die Maschinen mitsamt Daten vor 'dummen' Anwendern geschuetzt werden muessen und bestimmte Interpreteraktionen,

fuer alles moegliche, als zwangsweise Verknuepfung angenommen wird (das heisst dann 'Methode'), aber genauso wie der Absatz z.B. von Autos sich als schwierig erweist, die so etwas wie dreimaliges Haengenbleiben mit leerem Tank als generellen Totalausfall mit Hang zum Totalschaden quittieren, waechst wohl wieder die Bereitschaft, dass man die Fehlertoleranz mitsamt Freizuegigkeit im Umgang mit seinen Daten an den Anwender zurueckgibt. Das ist auch so ein Horror aus dem Kabinett des ideologischen Ueberbaus - Daten, die nurmehr zwangsweise interpretiert werden. Der naechste Schritt ist, natuerlich, zwangsweise plus limitiert. Aber auch das ist in der Schwebe und in der Tendenz unausgegoren, wo der Schuss losgeht.

<jomei!> waren das Zeiten, als man einfach nur so mal ein Programmerl abgefasst hat und sich freute, wenn es lief! Als ob das schon lange her waere.

Hannes Petersen wrote:

> Zusammen mit Precompilern, Recompilern und vielleicht
> noch Parsern ergibt sich ein wirklich schoener Sumpf von
> Begriffen, den viele Leute verschieden
> kompilieren und interpretieren.
> Ich halte einen Pascalcompiler fuer einen Compiler, auch
> wenn er (wie es mal vorgesehen war) P2-Code
> erzeugt, der dann interpretiert werden muss.

Ich hatte das in einem anderen Posting angefuehrt, dass da eine zeitliche Schicht der sprachlichen Verunsicherung dazwischen ist, mit auffaelliger Haeufung von akuten Zustaenden absurden Euphemismus'. Der Zeit verdanken wir diese Verunsicherung, nicht zu wissen, was zur Normalsprache gehoert und was zur Fachsprache, weil die Propagierung ueber die euphemistischen Kanaele auf Ueberanpassung getrimmt waren. Das hat sich ein Stueck abgehangen, und das Resultat eines Compilerlaufs ist CPU-nah und optimiert, jedenfalls ein Programm, und auch P-Code sieht aus wie ein Programm, weil man ja nicht hineinschauen kann, ausserdem wurde es von einem Compiler erzeugt. Ich denke nicht, dass da normalsprachlich dieser Sumpf noch vorhaelt. Der Nebel jener Apologeten ist zu einem guten Teil verraucht und um so viel wie sie sich wichtig vorkamen, ist dieser Stil nun doch eher unwichtig geworden.

Es ist wichtiger geworden, den gesellschaftlichen Bezug klar zu sehen. Dazu gehoert der Anspruch, dass Normalsprache greift. Und es daemmert wohl auch, dass das Erbe der Zeit der sprachlichen Schlamperei sich in Schlamperei mit Programmen fortsetzt und man sollte hoffen, dass der entsprechende Anspruch dort greift, wo diese Schnoddrigkeit von Imperatoren (wie sie mit genialer Miene ueber die Technikzukunft schwadronierten) sich im Zynismus von Programmgestaltern fortpflanzt, umso mehr zu spueren, als der analoge, haendische Ausstieg mehr und mehr eingespart wird, man solchem Zynismus, solcher Schoddrigkeit also ausgeliefert ist. Immer haeufiger ohne Ausweg, als ein dummes Raedchen in einem Algorithmus.

> Lange Rede kurzer Sinn: Deine Klassifizierung bzw
> Definition ist natuerlich richtig, wird aber von allen Seiten
> angefressen, sodass du bei manchen Gelegenheiten
> Radio Eriwan spielen musst. "Im Prinzip wird diese Sprache
> interpretiert, aber die Sprachbefehle wurden durch einen
> Precompiler in Token umgewandelt, Zahlen und Constanten
> in maschinenlesbarer Form gebracht, die Syntax komplett
> ueberprueft, ueberfluessige Programmteile entfernt,
> Variablen durchnummeriert, ...“

Interessant ist, was man zur Laufzeit einfuettert, weil das macht den Unterschied. Das ist auch normalsprachlich interessant. Der Rest ist Fachsprache. Wenn ich zur Laufzeit nicht alles einfuettern darf, was zum Repertoire gehoert, dann ist es auch nicht eigentlich ein Interpreter.

Martin Gerdes wrote:

>> Vielleicht ein springender Punkt ist hier
>> der rekursive Moment, wenn eine Sache sich selbst aus
>> dem Sumpf ziehen ...

> Hoppala: "das".

Hatte noch spekuliert, aber mit Echtzeit-Systemen im Hinterkopf passt mir auch der Moment als Augenblick. Das waere dann der gleichzeitige Verweis auf die Mehrdeutigkeit von 'Moment'. Aber, nun gut, ein rekursiver Zeitpunkt waere doch eher etwas von surrealistischen Ufern.

> Damals hatte Forth aus just diesem Grund eine gewisse
> Bluete.
> Mittlerweile duerften menschenverstaendlichere Sprachen
> zu Recht eine groessere Bedeutung haben.

Nein, _ich_ hab jetzt _nicht_ damit angefangen! ;)

Warum sich Forth so schwer mit der Akzeptanz tut, wird wohl ein Raetsel bleiben. Mit der Standardisierung (ANS '94 und spaeter ISO) ist es doch aufs Neue ein passables Arbeitsgeraet. Vielleicht hat es mit der UPN zu tun (s.u.), die steht quer zum Mathematikunterricht. Und dieserselbige ist ein gemeinsames Geruest aus dem common sense, mittels dessen sich billigerweise darauf abstellen laesst, dass alle Programmiererei direkter Abkoemmling der Mathematik sei.

In Forth merkt man sofort, dass das ein Schmarrn ist, weil man bei der Notierung ziemlich freie Hand hat und sie sich zurechtlegen kann, wenn man will. Vielleicht wird dies oefters einmal als harter Dorn im Nerv der Correctness in diesem Zirkus missverstanden.

Also wenn es sehr fachmaennisch aussieht, hinzuschreiben:

< Lampe_an{23,95) >

dann ist das Argument mit der Lesbarkeit zwar ueblich, aber eben nicht stimmig, weil man kann sich das zurechtlegen wie

man will und meist (dank UPN) schreibt man eher:

< gruenes Licht an blinkend >

und das war ein Kommando im Forth-Interpreter, nur sieht es ueberhaupt nicht mehr fachmaennisch abgehoben aus, vielleicht wurmt das einige. Man kann sich Spaesse erlauben, verglichen damit schauen die meisten Sprachen, die in der Tradition von Fortran und Algol stehen (Basic, Pascal, C usw.), eher wie Kaefige fuer Masochisten aus, wenn dort alles, was eine Maschine tun soll, aussehen muss, als ob eine Rechenaufgabe geloest wird (kraft algebraischer Notation, mit endlosen Zuweisungsketten als Alibi), wohingegen Forth-Quelltext einfach eine muntere Folge von weitgehend syntax-freien Nennungen ist (selbst eine Zahl ist nur eine einfache Nennung, die man reichlich kontextfrei hinschreiben kann und sie macht dennoch Sinn: der Wert harrt nun auf dem Stapel der weiteren Verwendung).

Wenn man nett ist, schreibt man, als ob eine Syntax noetig waere. Das macht man dann sich selbst zuliebe und nicht fuer ein externes Regelwerk. Man moechte sein Opus ja in drei Jahren auch noch lesen koennen, und weil das einige unterschaetzen, ruempfen denn wiederum andere darob die Nase und wenn dies den Arbeitsstil treffen sollte, so trifft es doch das Werkzeug.

Das verlangt viel Disziplin und das Frustrationspotential liegt eher darin, dass es kaum Vorschriften gibt, wie diese Disziplin auszusehen hat, und dann machen sich die Fallstricke eher noch empfindlicher bemerkbar als anderswo. Deshalb ist der Interpreter mit seinem ad-hoc-Compiler ein wichtiges Instrument, das in staendiger Uebung steht, um die Tragfaehigkeit kleiner Fragmente zu erproben, dieweil gestrickt wird (typischerweise in der Abstraktionsrichtung 'von unten nach oben'). Weswegen typischerweise (nachweislich) Forth-Programme schneller geschrieben werden und stabiler laufen als entsprechende Realisierungen im Mainstream (wo meist die Abstraktionsrichtung 'von oben nach unten' propagiert wird). Bessere Effektivitaet im Einsatz ist aber fuer gewoehnlich kein Argument im Software-Zirkus, ebenso wenig wie Sparsamkeit beim Umgang mit Ressourcen - solche ganz alltaegliche Irrationalitaet sieht man deutlicher wohl erst, wenn man sich die Erfahrung abseits des Mainstream gibt.

Die UPN, das ist eines der Details, die Disziplin erfordern, und dass man die UPN lernen muss, ist natuerlich ein Handicap, aber mit den HP-Taschenrechnern haben das auch viele freiwillig auf sich genommen und dann eher davon geschwaermt. Dass die meisten Daten im Quelltext keine Namen haben muessen (weil nur fluechtigerweise auf dem Stapel vorhanden), also Resultate eigentlich garnicht erwaehnt werden muessen, sondern nur die Operationen, die damit angestellt werden, verwirrt auch einige, wenn zugleich anderswo ein Zirkus um hierarchische Benamsungen abgehalten wird, mit inhaerenten Zuordnungen und automatischen Mutmaßungen und leicht noch mehr an solipsistischen Verrenkungen.

Besser lesbar wird auch dies damit nicht. Es lassen sich aber mehr Versprechungen mit dem Verkauf der Produkte verknuepfen, und das zaehlt.

(Wer zur UPN im Regen steht: Umgekehrt Polnische Notation [bzw. RPN, reverse polish notation] bezieht sich auf den polnischen Mathematiker Jan Lukasjewitz, der in den 1930er Jahren ein Aussagenkalkuel unterbreitet hat, fuer eine durchgaengig klammerlose Notation; zufaelligerweise harmoniert diese Notation nun sehr schoen mit dem Maschinenelement 'Stapel', auf dem sich am effektivsten Zwischenergebenisse ablegen lassen, ohne die Wiedereintrittsfaehigkeit von Code zu beeinflussen, also fuer einen stabileren Betrieb als wenn man dafuer Variablenplaetze naehme - weswegen in den meisten anderen Programmiersprachen ebenfalls die normale Schulnotierung intern erstmal in eine UPN-Entsprechung uebersetzt wird.)

Martin Gerdes wrote:

> Wenn man wenig Rechenleistung zur Verfuegung hat, ist
> eine Konstruktion wie Forth absolut genial. Die innerste
> Schleife, dauernd ausgefuehrt, besteht im Idealfall nur aus
> zwei Instruktionen (vorausgesetzt, der Prozessor hat eine
> selbstinkrementierende Adressierungsart). Dafuer aber
> hat man absolut keine Sicherheit in absolut keiner Art und
> Weise und hat den Rechner schneller abgeschossen, als man
> schauen kann.

Der 'innere Adress-Interpreter', die 'Next'-Maschine (und Du meinst wohl das klassische 'Next' mit den beiden Assemblerbefehlen fuer die 8086-CPUs, lods SI,AX; jmp AX), das ist eine logische Zwischenschicht, von der Idee her dasselbe nun in den diversen Virtuellen Maschinen, voruebergehend im P-Code. Forth-83 wurde damit definiert (mit 'Code-Feld-Address' usw.). Forth nach ANS bzw. ISO abstrahiert eine Ebene weiter, dem folgen viele der realen Forth-Interpreter. Der erzeugte Code hat oftmals wenig direkte Aehnlichkeit mehr mit dem, was man einfuettert. Dass die Instanz des Compilers abhebt, das hat auch hier Einzug gehalten. Das fruehere Prinzip ist genial zur Veranschaulichung, bestimmt auch verblueffend, wenn man es das erste Mal sieht, gehoert aber nicht mehr zu den Grundlagen.

> Heute ist weder Rechenleistung noch Speicherplatz ein
> Problem. Mir persoenlich sind Programme eine ganzes
> Stueck lieber, die nach Bereichsueberschreitung mit einem
> Laufzeitfehler auf die Nase fallen als welche, die gleich den
> ganzen Rechner platt machen.

Den macht aber das Betriebssystem mit Hilfe der Speicherverwaltung (mit MMU, memory management units). Wenn man das in Software haben will, kann man das ueberall einprogrammieren, das kostet aber empfindlich viel Rechenzeit. Was Du ansprichst, scheint aber der weitverbreitete Glaube zu sein, ein elaborierter Compiler koenne solche Zustaende vorausahnen. Das ist dann ein riskantes Spiel. Weil, bei nicht-trivialen alltaeglichen Programmen kann er das nicht.

Es ist aber eine nachvollziehbare Fortsetzung eines alten Spiels, und Rechner werden dann in barocker Manier als Abbild von mathematischem Denken aufgefasst[*]. Diese Projektion ist leider oftmals staerker, als dass man denn hinschaue, was eigentlich passiert. Und das ist eine ganz andere Schicht, als wenn im Positivismus Mathematik als generelle Methode postuliert wird. Dort weiss man, dass man sich Mathematik zunutze macht, als Werkzeug, das mit der Welt der Dinge eigentlich nichts zu tun hat. Bei jener Projektion geht hingegen die Annahme vorweg, dass sich mit betonierter Rechnerei die Mathematik irgendwie verdinglicht habe. Im Detail wird das korrigiert, wenn es sich als nicht haltbar erweist, aber in der normalsprachlichen Propagierung schlaegt das doch seine Kapriolen: Tiefstes Barock. [**]

Und die feisten Engelchen mit den glitzernden Posaunchen kuenden denn verklaert von der Unfehlbarkeit des kompilierten Quelltexts. Und der Chor der Brummbaerte von den Instituten stimmt mit ein. ;0

[*] Barock daran ist der Vorgang des Abbildens, des Vor-Sich-Hinstellens, der Scheidung der res extensa von der res cogitans, in der Folge die fortwaehrende Eroberung der res extensa, Bestaetigungen als Mittel gegen die Einsamkeit der res cogitans, was hinzugefuegt werden muss, so zwischenzeitlich Ansaetze schon wieder verschuettet werden, dem Leib-Seele-Dilemma beizukommen.

[**] Finde solche Art der Betrachtung kaum sonstwo und hatte auch nicht die Zeit, mich dem eigentlich zu widmen, so mag es ein wenig roh klingen.

Hannes Petersen wrote:

```
> Oliver Cromm wrote:
>>Josef 'Jupp' Schugt meinte:
>>
>>> COBOL:  [...]
>>> Pascal:  Inc(supporters_of_hypothesis)
>>> BASIC:  SupportersOfHypothesis
                = SupportersOfHypothesis + 1
>>> C:          supporters_of_hypothesis + 1;
>>
>>Die von Pascal gefaellt mir am besten, aber was soll's.
>
> Wie gefaellt dir:
>   (incf supporters_of_hypothesis)
>
> oder die Hardcore-Version:
>   (incf supporters-of-hypothesis)
```

Jetzt fehlt doch noch eine Forth-Version. Der Parameter ist auf dem Stack, der hat keinen Namen, wozu auch. Also:

< 1+ >
Das war's.
Salut...

Ewald Pfau

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 37, April 2003

PBM op AVR
Willem Ouwerkerk

Pulsbreitenmodulation zur digitalen Leistungsregelung für den AVR (z.B. AT90S2313). Software- und Hardware-Lösung, in AVR-ByteForth. Programmierwettbewerb (Einsendung bis 1.6.2003): Im Beispiel geht es nicht von 0% bis 100%. Man schaffe Abhilfe.

HTK-commando's in Forth
Albert Nijhof

Anfänger haben Schwierigkeit mit der "Umgekehrt Polnischen Notation" (Postfix-Schreibweise - ohne Klammern). HTK.FRT macht es möglich, die "huis, tuin en keuken-notatie", HTK, zu verwenden, also die Wald-und-Wiesen-Schreibweise (die Infix-Schreibweise - mit Klammern). Forth kann alles! Auch das. Nicht, dass es besser wäre. Nur so aus Spaß. Als Zugeständnis an den Anfänger. "Das Programm funktioniert. Man braucht es nur abzuschreiben und einzusetzen. Den richtigen Durchblick bekommt man aber erst, wenn man sich sowieso schon an die Postfix-Schreibweise gewöhnt hat - und HTK vergessen kann." (Der Autor.) Es wird ein Extra-Stack eingeführt. Zum Sammeln. Zulässige Klammer-Tiefe 12. Die herkömmliche Kommentar-Klammer in Forth stört dabei. Sie wird als Doppelklammer "((" neu eingeführt. Die Klammer selbst bekommt die infix-übliche Bedeutung. Die Schlussklammer muss als eigenständiges Forth-Wort betrachtet werden und erfordert einen führenden Zwischenraum. Das Forth-Wort HTK wird als Vorsatzwort verwendet und verleiht dem nachfolgenden Wort Infix-Eigenschaften. Interessant, was sich der Autor alles einfallen lässt - und mit einleuchtenden Beispielen belegt:
HTK + HTK - HTK * HTK /
Aber auch: HTK . HTK EMIT HTK > HTK < HTK = HTK <> HTK IF HTK THEN HTK AND HTK OR HTK XOR HTK WITHIN

Nr. 38, Juni 2003

Morse op het ATS-bordje
Albert Nijhof

Radio-Amateure müssen zu ihrer Prüfung auch morsen können. Einen Morse-Piepser auf die ATS-Platine zu montieren und über Forth einzubinden, ist Pfennigkram. Ein gutes Werkzeug zum Üben. Mit ASCII-Anzeige. Die Geschwindigkeit kann eingestellt werden.

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.

Die wenigen Beiträge zu unserem Mai-Treffen der SVFIG, die ich zu schreiben gedachte, verschwanden von meiner Liste, als ich Kevin Apperts Notizen bei http://www.forth.org/svfig/ sah. Es gibt nichts, was ich diesem Report hinzufügen könnte, außer vielleicht einige von Kevins Witzen und Witzchen und den beobachteten Reaktionen der 20 Teilnehmer.

Ich muß Kevin ein Kompliment machen für die ausgezeichnete Arbeit zu den Reporten aller SVFIG-Treffen seit April 2002, und dafür, daß er mich, nachdem ich das jetzt nun einmal herausgefunden habe, von meinem Kampf mit meinen eigenen begrenzten Reporten entlastet.

Friederich, ich zähle wenigstens 50 Briefe an die Vierte Dimension, seit unserem Gedankenaustausch in 1996, die meine Berichte von "Jenseits des großen Teiches" enthalten. Das scheint mir eine gute Anzahl zu sein, meine "Berichte" zu beenden und meine zukünftige Korrespondenz (die vermutlich weniger regelmäßig werden wird) anders zu benennen, wie zum Beispiel: "Lebenszeichen aus dem Silicon Valley". Aber das ist auch schon alles, wonach ich mit meiner Bitte um Rückmeldung frage. Ich frage Euch alle drei, welches nichttechnische Geschreibe meinerseits Euren Lesern am meisten nutzt, bevorzugt von geringerem Umfang oder seltener, weil ich mir ebenfalls wünsche, Dr. Beierlein die Übersetzungsarbeit zu erleichtern (besonders jetzt, da er auf den Swap-Drachen Obacht geben muß).

Fortsetzung
Seite – 22 –

Pseudo random generator
Willem Ouwerkerk

In einem Elektronikbuch in der Bibliothek gefunden: Schieberegister, höhere Bits geXORt nach Bit 1 rückgekoppelt. Aufruf an die Theoretiker der Forth-gebruikersgroep, die mathematische Erklärung nachzuliefern.

Lebendiges Forth
Lebenszeichen aus den USA

Hallo Freunde,
Ich denke, ich brauche eine kleine Rückmeldung zu diesem Brief. Ich habe schon vor einigen Stunden mein Laptop hochgefahren, aber mir fällt einfach nichts ein, was es wert wäre, Platz in Euren jeweiligen Magazinen einzunehmen *(Anm.: der Brief ist auch an den Editor der Forthwrite gegangen).* Auf der Suche nach Ideen habe ich mich ins Netz begeben und mir de facto jede Forth-Seite der ganzen Welt angesehen. Meine Güte, alles was irgend jemand über Forth und seine Teilgebiete zu wissen wünscht oder wissen muß, ist mit wenigen Mausklicks verfügbar.

Und eine Seite ist mit der nächsten verknüpft, in englisch ebenso wie auf deutsch, russisch, chinesisch und andere; jeweils im Format der jeweiligen Muttersprache.

(Englische Forth-Gesellschaft)

Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich)
Suchen Sie unsere Webseite auf:
www.users.zetnet.co.uk/aborigine/Forth.htm
Lassen Sie sich unser Neuzugangs-Gratis-Paket geben.
Der Mitgliedsbeitrag beträgt 12 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Beschleunigte Zustellung (Air Mail)
ins Ausland kostet 20 Pfund.
Körperschaften zahlen 36 Pfund,
erhalten dafür aber viel Werbung.

Wenden Sie sich an:

Dr. Douglas Neale
58 Woodland Way
Morden Surrey
SM4 4DS
Tel.: (44) 181-542-2747
E-Mail: dneale@w58wmorden.demon.co.uk

Aktualisierung der Titelliste

Die Titelliste, deren Hauptteil in den Heften 2/2003 bis 4/2002 erschien, wurde von Fred Behringer zusammengestellt. Sie enthält alle in unserer Zeitschrift "Vierte Dimension" jemals veröffentlichten Artikel - oder sollte sie jedenfalls enthalten. Fehlmeldungen bitte unter der E-Mail-Adresse behringe@mathematik.tu-munechen.de . Die Sachgruppen sind alphabetisch geordnet, die Titel innerhalb einer Sachgruppe nach dem Erscheinungsheft, innerhalb eines Heftes nach der Platzierung im Heft. Manche Artikel sind mehrfach, unter verschiedenen Sachgruppen aufgeführt. Editorials, Direktorials, Meldungen und Leserbriefe sind nur in Auswahl vertreten. Die hiermit präsentierte Aktualisierung enthält die Hefte 2/2002 bis 2/2003 einschließlich. Vorbild beim Gesamtunternehmen war die von Jenny Brien stammende Liste unserer englischen Forth-Freunde in der Forthwrite.

Thema	Autor	Heft	Titel
Algorithmen	Beuster, Bernd	02-2	Elementare Sortieralgorithmen
Algorithmen	Sala, Filippo	02-4	QSort mit Locals
Algorithmen	Sala, Filippo	03-1	IndexSort
Amerika	Vinerts, Henry	02-2	Neues aus FIG Silicon Valley
Amerika	Vinerts, Henry	02-3	Neues aus der FIG Silicon Valley
Amerika	Vinerts, Henry	03-1	Neues aus der FIG Silicon Valley
Anwendung	Zobawa, Klaus	03-2	Debugging mit Mixed Signal Oszilloscope am Beispiel I^2C-Bus
Anwendung	Rieger, Ewald	03-2	Triceps - zur Demonstration von Pick-and-Place-Aufgaben
Assembler	Noble, Julian	02-2	Ein Assembl(i)eraufruf (Teil 1), Übers. Behringer
Assembler	Bitter, Martin	02-3	pbForth 2.1.5 erschienen: pbAsm
Assembler	Noble, Julian	02-3	Ein Assembl(i)eraufruf (Teil 2), Übers. Behringer
Assembler	Noble, Julian	02-4	Ein Assembl(i)eraufruf (Teil 3), Übers. Behringer
Berichte	Bitter, Martin	02-2	Bericht aus dem Drachenrat!
Berichte	Moore, Chuck	02-4	Forth - die frühen Jahre (Übers. Prinz/Behringer)
Briefe	Behringer, Fred	02-2	Übersetzung für "Umsteiger"
Briefe	Bitter, Martin	02-3	Rechenfähigkeit des Universums
Briefe	Zobawa, Klaus	02-3	Küstenforth
Briefe	Kalus, Michael	02-3	EDV-Komplexität
Briefe	Hempel, Ralph	02-3	pbForth 2.1.3
Briefe	Paul, Ulrich	02-3	GPS-Empfänger
Briefe	Schöne, Rolf	02-3	Tower-Power (Lego-RCX)
Briefe	Prinz, Thomas	02-3	Rhein-Neckar-Gruppe
Briefe	Hempel, Ralph	02-3	pbForth 2.1.4, music on the RCX
Briefe	Bitter, Martin	02-4	Lee-Effekt
Briefe	Bitter, Martin	02-4	Chuck Moore live
Briefe	Bitter, Martin	02-4	BACKLASH
Briefe	Bitter, Martin	02-4	"Embedded" als PDF-Download
Briefe	Behringer, Fred	02-4	VD gut
Briefe	Sala, Filippo	02-4	ATARI
Briefe	Sala, Filippo	02-4	Keine Angst vor LOCALS
Briefe	Merkel, Joachim	03-1	Enigma
Briefe	Prinz, Friederich	03-1	WWW.SCULPTUR.DE
Briefe	Prinz, Friederich	03-1	Embedded
Briefe	Prinz, Friederich	03-1	MinForth
Briefe	Kretzschmar, R.	03-1	Was war für mich an Forth wichtig?
Briefe	Baecker, Gerard	03-1	Verteidigung der "Zunft der Informatik"
Briefe	Bitter, Martin	03-1	Dank an einen Forth-Programmierer
Briefe	Sala, Filippo	03-1	Errata zu QSort
Briefe	Sala, Filippo	03-2	Errata zu ISORT
Briefe	Paul, Ulrich	03-2	Zum Brief von Gerard Baecker
Briefe	Paul, Ulrich	03-2	Wettbewerb: Filser-Brief-Übersetzer
Briefe	Paul, Ulrich	03-2	Kleine Aufgabe
Briefe	Deliano, Rafael	03-2	Smartcards
Briefe	Strotmann, C.	03-2	Regeln zur guten Forth-Implementierung?
Briefe	Deliano, Rafael	03-2	Antwort an Carsten Strotmann
Briefe	Ertl, Anton	03-2	Antwort an Carsten Strotmann
Chatten	Bitter, Martin	02-4	Chuck Moore live

Compiler	Wilke, Jens	02-3 Instant (CrossCompiler-Vortrag)
Compiler	Schemmert, W.	03-1 Neues von "Avise"
Direktorial	Behringer, Fred	03-2 Aufruf: Mitgliederliste
Editorial	Prinz, Friederich	02-2 Liebe Leser, (Bitte um mehr VD-Beiträge)
Editorial	Prinz, Friederich	02-3 Liebe Leser, (Wo im Urlaub wird die VD gelesen?)
Editorial	Prinz, Friederich	02-4 Liebe Leser, (Dank an alle Mitwirkenden)
Editorial	Prinz, Friederich	03-1 Liebe Leser, (Begrüßung von 2 neuen Mitgliedern)
Editorial	Prinz, Friederich	03-2 Liebe Leser, (Junge Leute auf der Tagung)
England	Behringer, Fred	02-2 Forthwrite 109 (November 2000)
England	Behringer, Fred	02-2 Forthwrite 113 (September 2001)
England	Behringer, Fred	02-2 Forthwrite 115 (Januar 2002)
England	Behringer, Fred	03-1 Forthwrite 116 (April 2002)
England	Behringer, Fred	03-1 Forthwrite 117 (Juli 2002)
England	Behringer, Fred	03-1 Forthwrite 118 (September 2002)
England	Behringer, Fred	03-2 Forthwrite 119 (Januar 2003)
England	Behringer, Fred	03-2 Forthwrite 120 (März 2003)
Geschichte	Moore, Chuck	02-4 Forth - die frühen Jahre (Übers. Prinz/Behringer)
Geschichte	Moore, C.H. Rather, E.D.	03-2 FORTH-Progr. zur Beobachtung v. Spektrallinien (Übers. F. Prinz)
Grundlagen	Deliano, Rafael	02-4 Zyklische Codes
Grundlagen	Daneliuk, Tim	03-2 OO ist (k)eine Antwort (Übers. Friederich Prinz)
Hardware	Schöne, Rolf	02-2 Tower-Power
Hardware	Schleisiek, K.-P.	02-3 MicroCore, Vortrags-Zusammenfassung
Hardware	Krüger, Adolf Kalus, Michael	02-3 RCX am Draht
Hardware	Ouwerkerk, W.	02-3 Die Arbeitsgruppe Ushi (Vortrag)
Hardware	Paysan, Bernd	03-1 b16 - Ein Forth Processor im FPGA (Vortrag)
Hardware	Schemmert, W.	03-1 Neues von "Avise"
Holland	Behringer, Fred	02-2 Vijgeblaadje 29 (Dezember 2001)
Holland	Behringer, Fred	02-2 Vijgeblaadje 30 (Februar 2002)
Holland	Behringer, Fred	03-1 Vijgeblaadje 31 (April 2002)
Holland	Behringer, Fred	03-1 Vijgeblaadje 32 (Juni 2002)
Holland	Behringer, Fred	03-1 Vijgeblaadje 33 (August 2002)
Holland	Behringer, Fred	03-1 Vijgeblaadje 34 (Oktober 2002)
Holland	Behringer, Fred	03-1 Vijgeblaadje 35 (Dezember 2002)
Holland	Behringer, Fred	03-2 Vijgeblaadje 36 (Februar 2003)
Lernen	Beuster, Bernd	02-2 Assemblieren als Hobby?
Lernen	Noble, Julian	02-2 Ein Assembl(i)eraufruf (Teil 1), Übers. Behringer
Lernen	Noble, Julian	02-3 Ein Assembl(i)eraufruf (Teil 2), Übers. Behringer
Lernen	Noble, Julian	02-4 Ein Assembl(i)eraufruf (Teil 3), Übers. Behringer
Lernen	Deliano, Rafael	02-4 Zyklische Codes
Literatur	Behringer, Fred	02-2 Titelliste (Teil 1)
Literatur	Behringer, Fred	02-3 Titelliste (Teil 2)
Literatur	Behringer, Fred	02-4 Titelliste (Teil 3)
Lokale Daten	Sala, Filippo	02-4 QSort mit Locals
Meldung	Hoffmann, Ulrich	02-2 VD elektronisch
Meldung	Prinz, Friederich	02-2 Lycos mit Forth
Meldung	Prinz, Friederich	02-2 Enth
Meldung	Hempel, Ralph	02-2 pbForth 2.1.0, USB-Tower-Support
Meldung	Buß, Frank	02-3 Forth-Interpreter als Java-Applet
Meldung	Prinz, Friederich	03-1 MinForth
Meldung	Prinz, Friederich	03-2 Neues vom Forthbüro
Meldung	Schöne, Rolf	03-2 Forthbüro: Vorstellung
Mitteilung	Bitter, Martin	02-2 Laudatio auf den Drachenbewahrer

Vierte Dimension – Titelliste

Mitteilung	Schöne, Rolf	03-2 Forthbüro: Vorstellung
Mitteilung	Behringer, Fred	03-2 Aufruf: Mitgliederliste
OOP	Daneliuk, Tim	03-2 OO ist (k)eine Antwort (Übers. Friederich Prinz)
Philosophie	Beuster, Bernd	02-2 Assemblieren als Hobby?
Philosophie	Klimas, Andreas	02-2 Mein Weg zum Web-Anwendungsserver
Philosophie	Daneliuk, Tim	03-2 OO ist (k)eine Antwort (Übers. Friederich Prinz)
Philosophie	Daneliuk, Tim	03-2 Die beste Programmiersprache (Übers. Friederich Prinz)
Protokoll	Rieger, Ewald	02-4 Mitgliederversammlung 2002
Protokoll	Kalus, Michael	03-2 Mitgliederversammlung 2003
Prozessor	Paysan, Bernd	03-1 b16 - Ein Forth Processor im FPGA (Vortrag)
Prozessor	Schemmert, W.	03-1 Neues von "Avise"
Roboter	Schöne, Rolf	02-2 Tower-Power
Roboter	Krüger, Adolf Kalus, Michael	02-3 RCX am Draht
Roboter	Ouwerkerk, W.	02-3 Die Arbeitsgruppe Ushi (Vortrag)
Roboter	Rieger, Ewald	03-2 Triceps - zur Demonstration von Pick-and-Place-Aufgaben
Sortieren	Beuster, Bernd	02-2 Elementare Sortieralgorithmen
Sortieren	Sala, Filippo	02-4 QSort mit Locals
Sortieren	Sala, Filippo	03-1 IndexSort
Spaß	Prinz, Friederich	03-1 Richtigstellung (wie es wirklich mit Unix und C war)
Spaß	Prinz, Friederich	03-1 Wie Programmierer ihre Räder bauen
Spaß	Prinz, Friederich	03-1 Viruswarnung ?!?
Spaß	Daneliuk, Tim	03-2 Die beste Programmiersprache (Übers. Friederich Prinz)
Systeme	Ertl, Anton	03-1 Threaded Code - Varianten und Optimierungen
Systeme	Paysan, Bernd	03-1 b16 - Ein Forth Processor im FPGA (Vortrag)
Tagung	Prinz, Friederich	02-2 Bericht aus Garmisch-Partenkirchen
Tagung	Paysan, Bernd	02-2 Forth -Tagung 2002 in Garmisch-Partenkirchen
Tagung	Behringer, Fred	02-2 Stimmungsbild von der Forth-Tagung 2002
Tools	Nijhof, Albert	02-3 TO - ein Mechanismus mit vielen Möglichkeiten
Tools	Bitter, Martin	02-3 Ein WORDS für pbForth 2.1.3
Umsteiger	Noble, Julian	02-2 Ein Assembl(i)eraufruf (Teil 1), Übers. Behringer
Umsteiger	Beuster, Bernd	02-2 Assemblieren als Hobby?
Umsteiger	Noble, Julian	02-3 Ein Assembl(i)eraufruf (Teil 2), Übers. Behringer
Umsteiger	Noble, Julian	02-4 Ein Assembl(i)eraufruf (Teil 3), Übers. Behringer
Wettbewerb	Paul, Ulrich	03-2 Filser-Brief-Übersetzer

Ein Ingenieur, ein Mathematiker und ein Physiker sind beim Pferderennen.
Sie überlegen, ob es möglich ist, zu berechnen, welches Pferd gewinnt. Nach einer Woche treffen sie sich wieder.
"Ich habe überall nachgeschaut", meint der Ingenieur,"aber es gibt einfach keine Tabelle für Pferderrennen."
Der Mathematiker hat zwar bewiesen, das eine Formel existiert, er hatte aber nicht genügend Zeit, sie aufzustellen.
Der Physiker meint: "Ich habe eine Formel erstellt, mit der man exakt berechnen kann, welches Pferd gewinnt, sie hat allerdings einen Haken: sie gilt nur für reibungsfrei gelagerte, kugelförmige Pferde im Vakuum."

Der LINUX-Tag

Eine Anregung für die nächsten Jahre

Der SWAP-Drache unter Pinguinen FORTH auf dem Linuxtag, das war für manchen Besucher keine offensichtliche Verbindung. Bei genauerer Betrachtung gibt es einig Schnittpunkte zwischen der FORTH-Gemeinschaft und der OpenSource-Szene, schliesslich gibt es eine lange Tradition an FORTH-Systemen mit offenem Quellcode sowie eine Reihe guter FORTH-Implementationen für moderene, freie Unix-Systeme.

Der LinuxTag, die europaweit größte Messe rund um freie Software-Technologien, fand dieses Jahr zum zweiten Male im

Kongress- und Messezentrum in Karlsruhe statt. Wenn man den Zahlen der Veranstalter glauben kann, so konnte diese Veranstaltung entgegen dem allgemeinen Trend erheblich an Ausstellern und Besuchern zulegen. Die LinuxTag-Organisatoren, unterstützt durch Sponsoren, stellen nicht-kommerziellen Software-Projekten kostenlos die Infrastruktur für einen Messeauftritt zur Verfügung. Die FORTH-Gesellschaft hatte im Mai um einen Stand im Bereich der freien Projekte angefragt und einen schönen Platz neben dem Debian-Projekt, in der Nähe des Einganges, bekommen.

Im Laufe der vier Tage wurde hier von Mitgliedern der Gesellschaft über FORTH und die Aktivitäten der Gesellschaft informiert. Hauptattraktion des Standes, wenn nicht der ganzen Messe :-), war der Roboter von Ewald Rieger, den wir auch schon auf der FORTH-Tagung in Lambrecht bestaunen durften. Auf der Messe konnte der Roboter seine Rolle als Publikumsmagnet voll ausspielen, so daß zu jeder Zeit interessierte Besucher am Stand zugegen waren. Nach einer andächtigen Beobachtungsphase (je nach Besucher zwischen 5-15 Minuten) entwickelte sich meist ein Gespräch, mal kürzer, mal länger. Bei den FORTH-Unkundigen kam schnell die Frage "Was macht/ Wie funktioniert denn dieser Roboter? Was ist denn FORTH, und was hat das alles mit Linux zu tun?".

Eine zweite Gruppe von Besuchern reagierte in der Form "Ahh, FORTH. Das kenn' ich noch. Schön, daß es das noch gibt. Was wird denn heute mit FORTH gemacht?" Für beide Gruppen (und auch den Rest der Besucher) hatten wir Knoppix-Linux-CDs mit vier verschiedenen Linux-FORTH Versionen (BigFORTH, GNU-FORTH, FICL und LiNa) und einen zweiseitigen Informationszettel mit Informationen über FORTH und die FORTH-Gesellschaft erstellt. Beides fand guten Absatz, insgesamt wurden 40 CDs und über 150 Info-Zettel verteilt. Neben dem Roboter hatte Bernd Paysan den "b16"-FORTH Prozessor mitgebracht.

Dieser war nicht so schnell wie der Roboter für den flüchtigen Betrachter des Standes auszumachen, wurde aber, sobald entdeckt, mit viel Interesse betrachtet. Auf weiteren Rechnern wurden die BigFORTH OpenGL Demos und die OpenGL Simulation des Roboters gezeigt. Am Sonntagabend waren alle Beteiligten leicht erschöpft, aber froh, daß die Premiere der FORTH-Gesellschaft auf dem LinuxTag gut funktioniert hatte, und viele Besucher fuer FORTH interessiert werden konnten. Es bleibt zu hoffen, daß wir den einen oder anderen LinuxTag-Besucher in der Zukunft als Vereinsmitglied wiedertreffen können. Auf jeden Fall war dies ein gelungenes Experiment, welches auf dem LinuxTag 2004 wiederholt werden kann. Am LinuxTag-Auftritt haben mitgewirkt (in alphabetischer Reihenfolge):
Ullrich Hoffmann (Unterstuetzung, Web-Werbung, FORTH-Gesellschaft-Logo), Bernd Paysan ("b16" FORTH-CPU, Stand-Besatzung), Holger Petersen (Stand-Besatzung), Thomas Prinz (Fotos, Stand-Besatzung), Ewald Rieger (Roboter, Stand-Besatzung), Rolf Schoene, FORTH Buero (finanzielle Unterstuetzung), Martin "Joey" Schulze (unser Kontakt beim LinuxTag-Team), Carsten Strotmann (allg. Organisation, Stand-Besatzung).

Carsten Strotmann

pOOP in Forth

Alles ändert sich. Alles bleibt wie es war.

Beitrag zur FORTH-Tagung '03
vom 11. bis 13.4.2003 in Lambrecht
von
Manfred Mahlow
manfred.mahlow@anwind.de

Übersicht

1. Einleitung
2. Die Eigenschaften der pOOP Umgebung
3. Das pOOP Vokabular
4. pOOP in Beispielen
   4.1 Einfache Datentypen
      4.1.1 Der Datentyp cell
      4.1.2 Punkte mit globalem Farbattribut
      4.1.3 Punkte mit privatem Farbattribut
      4.1.4 Zeiger, nicht nur für Arrays
   4.2 Arrays
      4.2.1 Eine Basisklasse für Arrays
      4.2.2 Ein Array für cells
5. Zusammenfassung

1. Einleitung

pOOP steht für prelude-basierte objektorientierte Programmierung.

1997 habe ich auf der Forth-Tagung der FORTH-Gesellschaft das PRELUDE-Konzept vorgestellt, ein Konzept, daß es ermöglicht, Objekte und Methoden in einfacher und effizienter Weise durch implizite Kontextumschaltung zu verbinden. Das war ein erster Ansatz für pOOP in Forth. Neben einer kleinen Änderung des Forth-Interpreters waren nur wenige neue Wörter erforderlich. Die vertraute Forth-Syntax konnte unverändert beibehalten werden.

In den vergangenen Jahren habe ich das Thema in grossen Abständen immer mal wieder aufgegriffen und den pOOP-Ansatz nach und nach weiterentwickelt. Das Resultat ist eine OOP-Erweiterung für Forth, die ich nachfolgend kurz vorstellen möchte.

2. Die Eigenschaften der pOOP Umgebung

Die pOOP-Erweiterung stellt eine OOP-Umgebung mit folgenden Eigenschaften bereit:

* Einfach und effizient mit typischer Forth-Syntax

* Kapselung von Objekten und Methoden

* Einfache Vererbung (single inheritance)

* Polymorphismus (operator overloading)

* Statische Bindung von Objekten an Methoden
  (early binding)

* Dynamische Bindung von Objekten an Methoden
  (late binding) ist möglich

3. Das pOOP Vokabular

Die pOOP-Erweiterung fügt dem Forth-Wörterbuch die folgenden Wörter hinzu:

```
Vocabulary  FORTH  :
??   OOP

Vocabulary  OOP  :
THIS  FOR  OBJECT  METHOD
 ROOT  ..  CLASS

Class  ROOT  : ??  WORDS
 .S  ORDER  ;  FOR  OBJECT
 ALIGN  UNITS  ALLOT  CLASS
 SUPER  ..  UNSEAL  SEAL
 DEFINITIONS
```

4. pOOP in Beispielen

4.1 Einfache Datentypen

4.1.1 Der Datentyp cell

Der primäre Datentyp in Forth ist die Speicherzelle (cell), die aus 1 cells Adresseinheiten besteht:

```
only forth also oop definitions

\ Definition der Klasse
  class cell  cell definitions  [1]

\ Definition der Objekteigenschaften
  1 cells this allot  [2]

\ Definition von Methoden
  ' @  alias @   ( oid -- x )  [3]
  ' !  alias !   ( x oid -- )
  ' ?  alias ?   ( oid -- )
  ' +! alias +!  ( x oid -- )

\ Definition einer Initialisierungsmethode
  : init ( x oid -- ) this ! ;  [4]

  this seal  [5]   oop definitions
```

Nach dem Laden der o.a. Definitionen steht im Forth-System die neue Klasse cell zur Verfügung. Durch Eingabe von cell ?? <Enter> kann man sich den Klassenkontext (class search order) anzeigen lassen:

```
cell ?? <Enter>

Class Order : CELL  ROOT
Current     : OOP
------------------------------------------------------------------------
INIT  +!  ?  !  @
------------------------------------------------------------------------
?? WORDS .S ORDER ; FOR OBJECT ALIGN UNITS
ALLOT CLASS SUPER UNSEAL SEAL DEFINITIONS

ok
```

Die Suche im Klassenkontext CELL beginnt in der Wortliste der Klasse und endet in der geerbten Wortliste der Basisklasse OOP ROOT. [6]

4.1.2 Punkte mit globalem Farbattribut

Als Beispiel für die Anwendung von Objekten definieren wir eine Klasse zur Beschreibung von Punkten in der Ebene:

```
  only forth also oop definitions

\ Globale Farbfestlegung für Grafikelemente
  cell object color   0 color init  [7]

\ Definition der Klasse
  class point  point definitions
```

1 Erzeugt die neue Klasse cell und macht diese zum aktuellen Compilationskontext
Jede Klasse besitzt eine eigene und mindestens eine geerbte Wortliste. Die Wortlisten sind zu einer statischen Suchordnung verknüpft. Die Suche nach einem Wort beginnt immer in der eigenen Wortliste und wird in den geerbten Wortlisten fortgesetzt. Die letzte Wortliste der Suchordnung ist immer die Wortliste der Basisklassse OOP ROOT. Jede Klasse besitzt ausser dem Informationen über den Speicherplatzbe-darf ihrer Objekte. Dieser wird während der Klassendefinition festgelegt.

2 Legt fest, dass jedem Objekt der Klasse cell eine Speicherzelle zugeordnet wird .
Zur Laufzeit übergibt ein Objekt die Adresse dieser Speicherzelle als Objektidentifier (oid) auf dem Datenstack.

3 Wörter des Forth-Kerns können per Alias-Definition als Methoden der neuen Klasse sichtbar gemacht werden.

4 Weitere Methoden können als Colon-Definitionen hinzugefügt werden, z.B. eine Initialisierungsmethode, die in keiner Klassendefinition fehlen sollte.

5 Versiegelt die neue Klasse. Danach können der Klasse keine weiteren Wörter hinzugefügt werden. Eine Klasse wird automatisch versiegelt, wenn ein erstes Objekt der Klasse erzeugt wird oder wenn die Klasse ein erstes Mal vererbt wird.
Hinweis: Die Versiegelung kann mit `cell unseal definitions` wieder rückgängig gemacht werden.

6 Durch Eingabe von .. `<Enter>` kann ein Klassenkontext wieder verlassen werden.

```
\ Definition der Objekteigenschaften [8]
  cell   for x  \ x-Koordinate des Punktes
         for y  \ y-Koordinate des Punktes

 \ Definition von Methoden
  : @   ( oid -- x y ) >r@ this x @  r> this y @ ;
  : !    ( x y oid -- ) >r@ this y !  r> this x ! ;
  : ?    ( oid -- ) ." x=" dup this x ?  ." y=" this y ? ;
  : draw ( oid -- )
      this @  color @    \ x y color
    2drop drop ( Durch Zeichenfunktion ersetzen ) ;

\ Definition einer Initialisierungsmethode
  : init ( x y oid -- ) this ! ;

  this seal    oop definitions

\ Anwendung:  point object name   x y name init
\             name ?   name @   x y name !  name draw
\             name x ?  name y ?  usw.
```

Nach Laden dieser Klassendefinition wird durch Eingabe von point ?? <Enter> der Klassenkontext der neuen Klasse angezeigt:

```
Class Order : POINT  ROOT
Current      : OOP
------------------------------------------------------------------------
INIT  DRAW  ?  !  @  Y  X
------------------------------------------------------------------------
?? WORDS .S ORDER ; FOR OBJECT ALIGN UNITS
ALLOT CLASS SUPER UNSEAL SEAL DEFINITIONS

ok
```

4.1.3 Punkte mit privatem Farbattribut

Nehmen wir nun an, zu einem späteren Zeitpunkt werden Punkte benötigt, deren Farbe abweichend von der globalen Farbvorgabe festgelegt werden kann. Das läßt sich sehr einfach durch Vererbung lösen. Man definiert eine neue Klasse cpoint, die alle Eigenschaften der vorhandenen Klasse point erbt und weist den Objekten dieser neuen Klassse zusätzlich die Eigenschaft color zu:

```
  only forth also oop definitions


\ Definition der Klasse
    point class cpoint   cpoint definitions [9]

\ Definition der zusätzlichen Objekteigenschaft
    cell for color  \ Farbe des Punktes

\ Redefinition von Methoden
  : ? ( oid -- )
      dup this ? ." color = " this color ? ;
  : draw ( oid -- )
      this ? ( Durch Zeichenfunktion ersetzen ) ;

\ Redefinition der Initialisierungsmethode
  : init ( x y color oid -- )
      >r color @ r@ this color init
      r> this init ;

  this seal   oop definitions

\ Anwendung:  cpoint object name   x y color name init
\             name ?   name @   x y name !  name draw
\             name x ?  name y ?  name color ?  usw.
```

Nach Laden dieser Klassendefinition wird durch Eingabe von cpoint ?? <Enter> der Klassenkontext der neuen Klasse angezeigt:

```
Class Order : CPOINT  POINT  ROOT
Current      : OOP
------------------------------------------------------------------------
INIT  DRAW  ?  COLOR
------------------------------------------------------------------------
INIT  DRAW  ?  !  @  Y  X
------------------------------------------------------------------------
?? WORDS .S ORDER ; FOR OBJECT ALIGN UNITS
ALLOT CLASS SUPER UNSEAL SEAL DEFINITIONS

ok
```

4.1.4 Zeiger, nicht nur für Arrays

Arrays der gleichen Klasse können verschieden gross sein. Der Speichplatzbedarf von Arrays ist also keine Klassen-, sondern eine Objekteigenschaft. Es ist deshalb sinnvoll, Arrays als Zeiger zu realisieren und die Grösse der einzelnen Arrays erst während der Objektinitialisierung festzulegen.

Wir brauchen also einen Datentyp *Zeiger* (pointer):

```
only forth also oop definitions


\ Definition der Klasse
  class pointer  pointer definitions

\ Definition der Objekteigenschaften
  1 cells this allot

\ Definition von Methoden
  : @ ( oid -- addr ) @ dup 0= abort" uninitialised pointer" ;
```

7 Erzeugt und initialisiert ein `cell`-Objekt mit dem Namen `color` im Kontext `OOP`.
8 Die beiden Koordinaten eines Punktes werden auf `cell`-Objekte abgebildet.
9 Die Klasse `cpoint` erbt hier alle Eigenschaften der Klasse `point`.

```
\ Definition einer Initialisierungsmethode
 : init ( x oid -- )
    dup @ abort" can't reinitialise pointer" ! ;

 this seal   oop definitions

\ Anwendung:  pointer object name   addr name init   name @
```

4.2 Arrays

4.2.1 Eine Basisklasse für Arrays

Alle Arrays haben gewisse Gemeinsamkeiten. Es ist deshalb sinnvoll, eine Basisklasse zu definieren, von der alle anderen Array-Klassen erben können:

```
 only forth also oop definitions

\ Definition der Klasse

 pointer class array   array definitions  [10]

\ Definition weiterer Objekteigenschaften
 cell for size  [11]

\ Definition der Initialisierungsmethode
 : init ( u oid -- )
    here over this init  over allot  this size ! ;

\ Definition weiterer Methoden
 : of? ( i oid -- ? )
     >r dup 0 <  swap r> this size @ >= or ;
 : of ( i oid -- addr )
    2dup of? abort" index out of range " this @ + ;
 : size ( oid -- u ) this size @ ;

 this seal   oop definitions

\ Anwendung:  array object name   u name init  [12]
\    name @  [13] name size  [14]    i name of  [15]
```

4.2.2 Ein Array für cells

Unter Verwendung der Basisklasse *array* können beliebige andere Array-Klassen definiert werden. Am Beispiel eines cell-Arrays sei das nachfolgend gezeigt.

```
 only forth also oop definitions

\ Definition der Klasse
 array class cells()   cells() definitions  [16]

\ Definition der Initialisierungsmethode
 : init ( u oid -- ) >r cells r> this init ;

\ Definition weiterer Methoden
 : of ( i oid -- oid[i] ) >r cells r> this of cell ;  [17]
 : size ( oid -- u ) this size 1 cells / ;
 : fill ( x oid -- )
    dup this size 0 ?do  2dup i swap this of !  loop  2drop ;
 : ? ( oid -- )
    dup this size 0 ?do i dup ." : " over this of ? cr   loop
   drop ;

 this seal   oop definitions

\ Anwendung:  cells() object name   u name init  [18]
\                 i name of
```

5. Zusammenfassung

Mit den vorstehenden Beispielen sollte ein erster Einblick in die Möglichkeiten der pOOP in Forth gegeben werden. Die Beispiele liessen sich mit wachsender Komplexität endlos fortsetzen. Ich hoffe, es ist deutlich geworden, wie elegant, einfach und übersichtlich Klassen definiert und Objekte und Methoden angewendet werden können. Der Umgang mit komplexen Datenstrukturen wird so sehr einfach.

Der Schlüssel zu dieser Einfachheit ist die implizite Kontextumschaltung zwischen der search order des Forth-Kerns und der klassenspezifischen class search order der Klassen:

* Ein Objekt ersetzt die aktuelle search order durch die class search order seiner Klasse.

* Eine Methode ersetzt die aktuelle search order durch die search order des Forth-Kerns oder durch eine andere ihr zugewiesene class search order.

Die Kontextumschaltung erfolgt zur Interpretations- oder Compilationszeit (early binding). [19]

10 Die Klasse `array` erbt alle Eigenschaften der Klasse `pointer`.
11 Die Grösse des Arrays in Adresseinheiten (units). Sie wird für jedes Array-Objekt bei der Initialisierung gesetzt.
12 Reserviert *u* Adresseinheiten Datenspeicher für das neue Array.
13 Übergibt die Adresse des Array-Datenspeichers auf dem Datenstack.
14 Übergibt die Grösse des Array-Datenspeichers in Adresseinheiten auf dem Datenstack.
15 Übergibt die Adresse der i-ten Adresseinheit des Array-Datenspeichers auf dem Datenstack.
16 Die Klasse `cells()` erbt alle Eigenschaften der Klasse `array`.
17 Macht die Klasse `cell` zum aktuellen Suchkontext (class search order) und übergibt den Objektidentifier des Array-Elements i auf dem Datenstack.
18 Reserviert *u* Speicherzellen für das `cell`-Array name.
19 Dynamische Bindung von Objekten mit Methoden (late binding) kann erreicht werden, indem man die entsprechenden Wörter erst zur Laufzeit interpretiert, z.B. `s" object-name method-name" evaluate .`

Durch die pOOP ensteht in Forth eine ganz neue Ebene der Faktorisierung. Analog zur Zerlegung von Wörtern in Wörter können Objekte in Objekte zerlegt werden. Oder, anders herum betrachtet, es können neue Objekte unter Wiederverwendung bereits definierter Objekte definiert werden.

Manfred Mahlow

MicroCore Philosophy

Klaus Schleisiek
zur Tagung der Forthgesellschaft in Lambrecht

MicroCores oberste Priorität ist Einfachheit und Verständlichkeit. MicroCore hat seine Wurzeln in der Forth-Programmiersprache, ist aber nicht auf die Ausführung von Forth-Programmen beschränkt - er ist ein recht guter Universalprozessor. Das Hinzufügen einer indizierten Adressierung in den Return-Stack erlaubt die einfache Übersetzung von effektiv ablaufenden C-Programmen.

Die Herangehensweise an seinen Entwurf unterscheidet sich jedoch von den meisten anderen Prozessor-Kernen: Der Assembler war zuerst vorhanden und er realisierte ungefähr 25 Forth-Primitives. Während andere Prozessoren versuchen, die größte Befehlsvielfalt aus einem Minimum an Hardware zu erzielen, findet man einige extrem spezialisierte Befehle hohen semantischen Inhalts in MikroCore, weil rund 30 Jahre Erfahrung mit Forth bewiesen haben, daß dies nützliche Primitives sind. Ungeachtet dessen wird jeder Befehl in einem Taktzyklus ausgeführt.

MicroCore ist nicht die einzige Architektur, die Forth als Assembler benutzt. Ich nahm Chuck Moores NC4000 als Modell und erweiterte diesen zum FRP1600, der niemals über das zweite Silizium und seinen Interrupt-Bug hinauskam. Ein frischer Ansatz war die Realisierung des "Feldbus Prozessors" IX1, der sich nach wie vor in der industriellen Automation verkauft. Er führte die Harvard-Architektur bei den Forth-Maschinen ein. Es ist möglich, daß die MicroCore-Architektur aus einigen Diskussionen mit Christophe Lavarenne über die Transputer Architektur resultiert. Er nutzt zwei Erfindungen des Transputers; nämlich: Die Aneinanderkettung von "Nibbeln" um Literale höherer Genauigkeit zu formen, die von nachfolgenden Befehlen konsumiert werden. Dies ist die Schlüsseltechnologie, die MicroCore's Objektcode unabhängig von der Datenbusbreite macht. Weiterhin sind es TRAP-Signal und -Befehl als Hardwaremechanismus zur Behandlung der Ressourcenverteilung, als Schlüsseltechnik für effizientes Multi-Tasking.

MicroCore versucht bezüglich des Verhältnisses von Hardwarekomplexität zu Befehlssemantik ein Optimum zu erreichen. Es gibt einfachere Architekturen, allerdings auf Kosten der Befehlssemantik. Aber dann muß man mehr von diesen einfacheren Befehlen ausführen, um eine bestimmte Aufgabe zu erledigen. Dies führt jedoch schließlich zu einem höheren Energieverbrauch, da das Lesen der Befehle aus dem Programmspeicher üblicherweise die Hauptquelle des Energieverbrauchs in Mikroprozessorsystemen ist. Darum versucht MicorCore die Dinge in Hardware zu erledigen, die am besten in Hardware erledigt werden, und überläßt die Operation der Software, die in Software effektiver und mit geringerer Komplexität getan werden. Um eine Vorstellung davon zu geben, was ich damit meine: vergleiche periphere Mikroprozessorkomponenten, die von Intel- und Motorola-Ingenieuren entworfen wurden: Intel hat wiederholt versucht, Softwareprobleme in Hardware zu lösen, während Motorola es i.d. R. richtig hinbekommen hat.

Dank Forths Erweiterbarkeit - d.h., die Offenheit seines unkonventionellen Compilers für Modifikationen und Erweiterungen durch den Anwender - ist der Standardisierungsprozeß für Forth ein langsamer, Konsens bildender Prozess in einer großen Gemeinschaft, verglichen mit den Spezialistenkreisen der Compilerschreiber bei den meisten anderen Sprachen. Forth-Worte können als Mikrozellen eines Softwaresystems gesehen werden, welches gebildet wurde, um ein spezifisches Problem zu lösen. Dies ist die gleiche Weise in der MicroCores Mikrozellen genutzt werden sollten, um ein komplexeres, spezifisches Stück Hardware zu bilden. Einfachheit ist ein Anzeichen geeigneter Faktorisierung, und Verständlichkeit ist die Bedingung für eine breite Akzeptanz. Geeignete Faktorisierung ist eine Sache der Erfahrung und des Verlangens nach ästhetischen Lösungen. Z.B. können 'geeignete' Baussteine eines UART uart_clk, uart_tx und uart_rx sein. Nur die Zeit und die Erfahrung kann uns das lehren.

Durch die Übergabe von MicroCore an die Öffentlichkeit, hoffe ich, daß er ein Katalysator für weitere periphere Funktionen, der Mikrozellen, sein wird, die unter den gleichen Lizenzbestimmungen plaziert werden. Als ursprünglicher Entwickler von MicroCore reserviere ich mir das Recht, die Konformität der von MicroCore abgeleiteten Arbeiten mit dem Originalcode zu zertifizieren. Dies ist mein einziges geschäftliches Interesse an MicroCore, neben der Verwendung für eigene embedded systems Entwicklungen. Diesbezüglich sind MicroCore's Lizenzbestimmungen von GPL oder anderen "Open Source" Lizenzen verschieden, da MicroCore sich mit Hardware befaßt, die Applikationscode abarbeitet, anstelle von Applikationssoftware oder den Werkzeugen zur deren Entwicklung.

Solange ich MicroCore nicht als Produkt unterstützen kann, werde ich keine "Public" MicroCore Lizenz anbieten. Stattdessen erlaubt die MicroCore Exploratory License" einzelnen Personen, Universitäten und Forschungsinstituten, mit MicroCore zu experimentieren. Ich werde eine "MicroCore Unilateral License" jeder Firma, die MicroCore unter der Exploratory License benutzt kostenlos erteilen, solange mich ihre Experimente nicht tangieren.

WWW.MICROCORE.ORG wurde zur Bildung einer Nutzergemeinde eingerichtet. *Klaus Schleisiek*

Meine Gratulation und meinen Dank an Dr. Beierlein.

Ich möchte ein wenig übermütig schließen. Wenn Ihr die Webseite der SVFIG besucht und Kevins Report zu dem Treffen im April 2003 lest, dann werdet Ihr sehen, daß die FIG noch nicht tot ist. Sie scheint gerade in der SVFIG aufzugehen, aus der alle neuen "Offiziellen" stammen - Präsident George (Perry, nicht W.) und drei unserer überlebenden Johns, wie V.P., Sekretär, und der Schatzmeister ebenso. Wir hoffen, daß wir einige der Früchte der Arbeit unseres früheren Präsidenten Carter (Skip ist gemeint) bewahren können.
Viel Glück Euch allen. *Henry.*

Liebe Leser,
für die Forthgesellschaft haben die Redaktion der VD und Fred Behringer unserem Reporter in Übersee Mut zu machen versucht. Wir haben versucht, dem Henry deutlich zu machen, daß gerade seine "nichttechnischen" Berichte uns immer wieder gezeigt haben, wie unsere Forth-Freunde in den USA Forth sehen, erleben und oft regelrecht genießen. Wir hoffen sehr, daß uns das gelungen ist. Henrys Berichte möchte niemand missen. Vielleicht sagen Sie ihm das aber am besten selbst. Sie erreichen Henry am einfachsten per Mail:
VOLVOID@AOL.COM

fep

Ich grüße Euch alle!

Ich möchte alle Leser wissen lassen, daß ich, als der "älteste Forth Novize" nicht länger versuchen werde, technische Berichte über die Treffen der SVFIG zurecht zu fummeln. Dave Jaffe (unser Webmaster) und Kevin Apert (unserer leidenschaftlichster Web-Surfer, wenn ich das so sagen darf) geben mit ihrer eigenen Notizen im Netz eine ausgezeichnete Arbeit ab. Zu dem Juni-Treffen könnt Ihr euch bei http://www.forth.org/svfig/kk/06-2003.html bestens informieren. Vielleicht erforscht Ihr dann gleich ein wenig von dem vielfältigen Material, das in den von dieser Web-Seite ausgehenden Verknüpfungen angeboten wird.

Nach der Ermunterung zu urteilen, die ich von Dr. Behringer, Friederich und Chris bekommen habe, scheint es, als sollte ich mit einer Art "Mini-Berichten" von dieser Seite des großen Teiches fortfahren, und sei es nur um Euch wissen zu lassen, daß Forth die Substanz ist, die eine gewisse Anzahl lebendiger Menschen einmal im Monat im Silicon Valley zusammenruft, wo sie über ihre Projekte reden können, miteinander vis-a-vis schwatzen können, Witze austauschen und einander beim Älterwerden beobachten (vielleicht beim weiser werden?). Ich hoffe, ich kann Euch "forthige Lebenszeichen" schicken, solange ich solche Phänomene beobachten darf.

Oh ja, ich habe rund 20 lebendige Körper während unseres Juni-Treffens gezählt, obschon nicht alle alle Zeit anwesend oder wach waren. Die Gründe, warum man zu den Treffen kommt, sind bei jedem Einzelnen andere. Aber solange Forth der gemeinsame Nenner ist, und solange dort geatmet wird, ist das für ein Hinweis auf ein lebendiges Forth.

Paradoxerweise hat Dr. Ting als erstes den Morgen mit einem Vortrag über C begonnen, genauer über C++, wie er selbst feststellte. Er hat sich dann aber sehr beeilt, uns zu erklären, daß dies für Ascorbinsäure (Vitamin C) stehe, möglicherweise seit 5.000 Jahren DAS Heilmittel für die meisten menschlichen Gebrechen, ähnlich der "magischen Kugel" oder dem "CHI", nach dem die Chinesen 5.000 Jahre gesucht haben. Mit der ihm eigenen Gründlichkeit und Ausdauer hat er solange viele Bücher und Veröffentlichungen studiert und geforscht, bis er sein Buch mit dem Titel "SARS, Kardioviskulare Erkrankung, und Vitamin C" veröffentlichen konnte. Keine Arbeit mit Forth während eines Monats; weil das Board und der Computer nicht mitgespielt haben. Ein interessanter Vortrag, von einem qualifizierten Vortragenden mit einem Doktorgrad in Chemie. Und jetzt beeilt sich vielleicht jemand, Vitamin F zu enträtseln.

Al Mitchell kam mit seinen "Geräten", verschiedenen Versionen von Cygnal's C8051Fxx Single-Chip Systemen, seinen Forth-Briefmarken, die er in verschiedenen Bausätzen vermarktet, gebrauchsfertig für den Massenmarkt, ähnlich den BASIC-Briefmarken. Das BASIC ist in Forth geschrieben, und Al hofft, mit den offenen Quellen wenigstens einige Nutzer auf Forth aufmerksam machen zu können. Al lädt dazu ein, Ideen zu Applikationen für sein System beizusteuern (siehe auch http://www.amresearch.com).

Al hat auch eine online-Version der letzten Forth-Dimensions erstellt, die nicht veröffentlicht wurde (Vol. XXI, 1,2, in 1999), und die man als FD.ZIP von seiner Web-Seite herunterladen kann (oder aber durch eine Verknüpfung von Kevin's Notizen auf der SVFIG Web-Seite).

Ihr seht, daß mein Brief schon allein dann zu lang wird, wenn ich nur zu beschreiben versuche, was alles im Netz an Interessantem für die Leser der VD und der Forthwrite zur Verfügung steht. Vielleicht sagt mir jemand einfach, womit ich meine Zeit besser nutzen könnte?
So viel für heute, Leute. Bleibt gesund und fröhlich.

Henry

Hallo, meine Freunde jenseits des großen Teiches!

Trotz des heißen Sommers, der einen von einer Fahrt ins Silicon Valley in einem 32 Jahre alten Volvo Kombi ohne Klimaanlage zurückschrecken läßt, habe ich es geschafft, an den Treffen im Juli und im August teilzunehmen, wenn auch nur jeweils für einen halben Tag (die erste Hälfte). Die erste Hälfte offenbart gewöhnlich eine Anzahl zu spät Kommender, einige davon übermüdet, weil sie in der letzten Nacht noch lange an ihren Projekten gearbeitet haben.

Eher mehr als weniger ist es der unermüdliche Dr. Ting, der die ersten beiden Stunden ohne Unterbrechung mit Vorträgen ausfüllt, die eine Vielzahl von Themen um Forth herum abdekken. Nach dem Mittagsessen ist das Treffen weniger strukturiert. Spontane Präsentationen, vielleicht bewußt erst gegen Ende serviert, leiten über in eine Atmosphäre von Diskussionen in Gruppen, Informationsaustausch und einem Hin und Her von Witzchen.

Im Juli hat Dr. Ting uns von FML (Forth Mark-Up Language?) erzählt, die in Taiwan für einen chinesischen Web-Browser entwickelt wurde. FML ist in DSForth (aus Rußland) geschrieben. Das interpretiert Forth-Worte, die in HTML Tag-

Klammern plaziert sind. Wie ich es verstanden habe, wird sich das System weiterentwickeln und Colon-Definitionen interpretieren, die in chinesisch geschrieben sind und sich ebenfalls innerhalb der Klammern befinden.

Im August hat sich Tings Vortrag mit den Grundlagen des Zeichnens von Linien und mit Polygon-Füllalgorithmen (für Hardware) auseinander gesetzt. Für das Füllen galt eine Effizienzrate von zwei Takten pro Pixel; jedenfalls besser als das Füllen durch das Zeichnen horizontaler Linien. Dabei habe ich auch gelernt, daß im Chinesischen die Hälfte aller Striche horizontal verlaufen.

Die Beteiligung am vormittäglichen Teil der Treffen war geringer als gewöhnlich. Knapp ein Dutzend lebender Körper, überwiegend vom harten Kern; aber sie waren ein Hinweis auf ein lebendiges Forth! Der Sommer hat ihren Geist nicht welken lassen.

Bis zum nächsten Mal,
bleibt cool!
(Es tut mir Leid, Dr. Beierlein, "cool" ist im amerikanischen Englisch doppeldeutig).

Henry

Nun, lieber Henry, da ich dieses Mal die Übersetzung angefertigt habe, hoffe ich, daß die Temperaturen des Sommers sich auch auf Eurer Seite des großen Teiches auf ein deutlich cooleres Maß gesenkt haben.

Fritz

Was ist Pflock-Solitaire?

Ewald Rieger
ewald.rieger@t-online.de

Hintergründe zum Triceps-Roboter

1. Das Spielfeld

Das original Pflock-Solitaire wird auf einem Brett mit 33 Löchern gespielt. In die Löcher werden 32 Pflöcke gesteckt, der mittlere Platz bleibt frei.

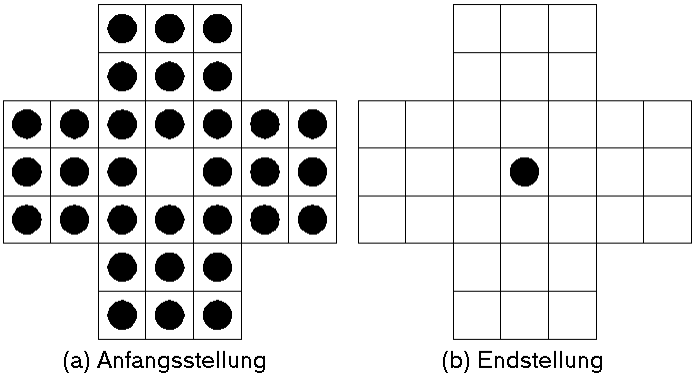


Abb.: 1 – Das Spielfeld

2. Wie spielt man Pflock-Solitaire?

* Jeder Zug besteht aus einem horizontalen oder einem vertikalen Sprung über einen benachbarten Pflock. Der übersprungene Pflock wird vom Brett genommen. Diagonale Sprünge sind nicht erlaubt!

* Das Ziel ist möglichst viele Pflöcke vom Brett zu entfernen.

* Gewonnen hat derjenige, bei dem nur ein Pflock in der Mitte stehen bleibt.

3. Historischer Rückblick

* Die Herkunft des Spiels ist nicht bekannt.

* Der Volksmund behauptet, daß ein gefangener französischer Adliger dieses Spiele während seines langen Aufenthalts in seiner Gefängniszelle erfunden hat.

* Im Jahre 1697 wurde das Spiel auf einem Portrait von Prinzessin Madame de Soubize abgebildet.

* Gottfried Leibniz (1646-1716) war die treibende Kraft zur Entwicklung von Regeln und Theoremen über das Solitairespiel.

4. Fragen zu Solitaire

* Wieviele Zustände gibt es auf dem Brett?

* Und wieviele Wege führen zu diesen Zuständen?

* Gibt es Mathematik, um Solitairespiele zu analysieren und zu entscheiden, ob sie lösbar sind.?

5. Fakten

* Pflock-Kombination: 2^33=8.589.934.592

* erreichbare Kombination: 23.475.688

* Anzahl der Spielzüge: 577,116,156,815,309,849,672
(577 quintillion 116 quadrillion 156 trillion 815 billion 309 million 849 thousand 672)

* Ein Supercomputer der 1 Billion Spielzüge pro Sekunde berechnen kann, müßte über 18000 Jahre rechnen.

* Wir brauchen einen besseren Algorithmus!

6. Auf der Suche nach Lösungsansätzen

* Yahoo! ergibt allein über 300 Treffer für "Peg Solitaire"

* Das Ergebnis zu Erreichbarkeits-Tests: Kann ein Test nicht vollständig durchlaufen werden, gibt es keine Lösung zu diesem Spiel. Wird ein Test durchlaufen, sagt er uns nichts.

* Gruppen-Theorie [1]

* Pagode Funktion

* Lattice Criterion

* "Finalist" arguments

* Pakete und Kahlschläge [2]

1 Arie Bialostocki in The College Mathematics Journal (3 May 1998)
2 Elwyn R. Berlekamp, John H. Conway, & Richard K. Buy in Winning Ways for Mathematical Plays. Kapitel 23

7. Was können wir tun, um zu gewinnen?

7.1 Pakete und Kahlschläge

Ein Paket ist eine Anordnung von Pflöcken, um sie mit Hilfe eines Katalysator-Pflocks zu schlagen.

* Dreier-Kahlschlag

* Sechser-Kahlschlag

* L-Kahlschlag

* Zweier-Kahlschlag

* Zweier-Paket

* Vierer-Paket

* L-Paket

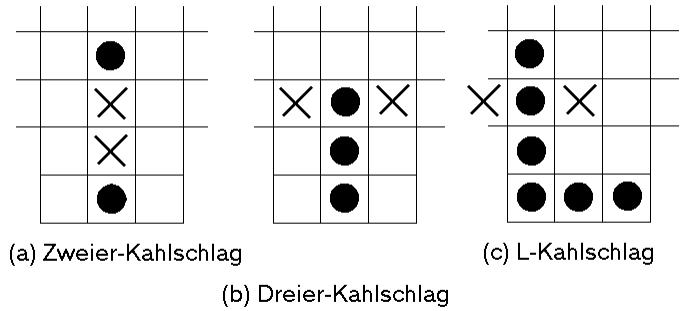


Abb.: 2 – Kahlschläge

7.2 Hilfsquellen

Hilfsquellen sind eine geschickte Kombination aus der Pagode-Funktion und einer Funktion zum Ausgleich zwischen den Zügen in Nord-Süd-Richtung und den Zügen in Ost-West-Richtung.

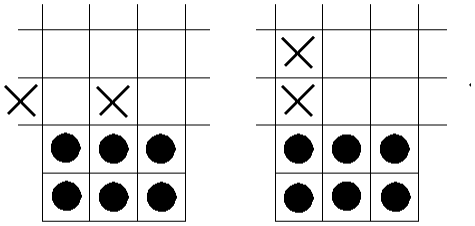


Abb.: 3 – Sechser-Kahlschläge

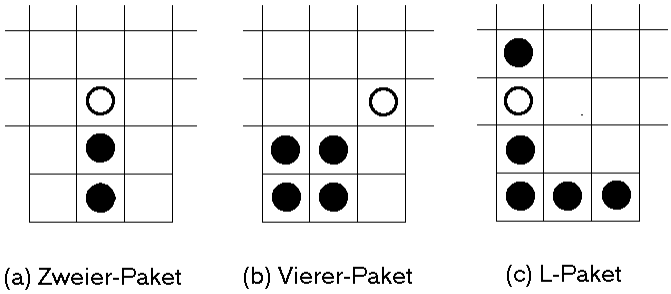


Abb.: 4 – Pakete

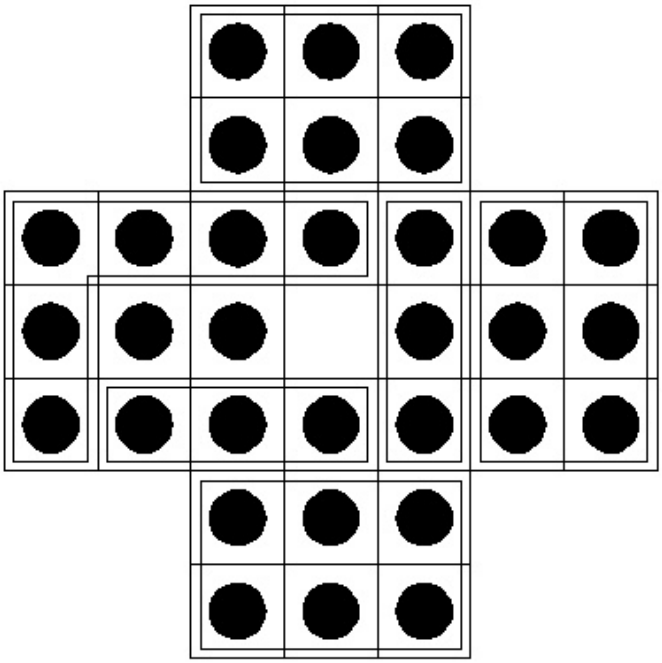


Abb.: 5 – Pakete zur Lösung des Zentralumkehrspiels

7.2.1 Das Bilanzbrett

Das Bilanzbrett ordnet jedem Pflock auf dem Brett einen Wert zu.

		b^{-1}	β	$b^{-1}\beta$		
		b	$a\beta$	$b\beta$		
a^{-1}	a	1	a	1	a	a^{-1}
α	$b\alpha$	b	$c\alpha\beta$	$b\beta$	$b\alpha\beta$	α
$a^{-1}\alpha$	$a\alpha$	1	$a\alpha$	1	$a\alpha$	$a^{-1}\alpha$
		b	$a\alpha\beta$	$b\beta$		
		b^{-1}	β	$b^{-1}\beta$		

Abb.: 6 – Bilanzbrett

7.2.2 Produktwerte der Züge

Jedem Zug kann ein Produkt-Wert zugeordnet werden, in dem wir das Produkt der beiden algebraischen Werte von den zwei beteiligten Pflöcken bilden und durch den Wert des Platzes, auf dem der Pflock landet, dividieren. Nachfolgende Gleichung zeigt ein Beispiel für den ersten Zug beim Zentralumkehrspiel von West nach Ost.

$$\frac{b\alpha * \beta}{c\alpha\beta} = b^2c^{-1}\beta$$

Bei allen Berechnungen ist die Relation $a^2 = b^2 = \mathbf{1}$ zu beachten.

Die nachfolgende Tabelle zeigt die zehn unterschiedlichen Werte, die Züge in einem Spiel annehmen können.

a	$a\alpha$	$c\alpha$	$a^2c^{-1}\alpha$	a^2	b	$b\beta$	$c\beta$	$b^2c^{-1}\beta$	b^2

7.2.3 Hilfsquellen eines Spiels

Produktwerte können auch über mehrere Züge oder ein ganzes Spiel gebildet werden. Dabei wird das Produkt der Ausgangsstellung durch das Produkt der Endstellung dividiert. Für das Zentralumkehrproblem ergibt sich folgender Wert.

$$Hilfsquellen = \frac{a^4b^4}{ca\beta} = a^4b^4c^{-1}\alpha\beta$$

Weiter gilt, daß bei einem erfolgreichen Spiel dieses Hilfsprodukt exakt durch seine Züge verbraucht werden muß. Dazu wird Zug für Zug, das noch vorhandene Hilfsquellenprodukt durch den Wert des gerade gespielten Zuges dividiert.

7.3 Kombinatorik der Hilfsquellen

Zur Bewertung eines Spiels bilden wir alle möglichen Kombinationen aus Zugwerten, die das Hilfsprodukt aufbrauchen und merken uns in einem Array, wie oft ein bestimmter Zug in der Kombination vorkommt. Die sich aus der Berechnung ergebende Reihenfolge der Züge ist willkürlich und interessiert uns nicht weiter.

7.4 Auf der Suche nach einem Weg

Im ersten Schritt berechnen wir die Zugkombination aus den Hilfsquellen eines Spiels. Ausgehend von der Anfangsstellung besuchen wir in einer Tiefenssuche alle Züge für die es einen Zugwert in der errechneten Kombination gibt. Jeder ausgeführte Zug wird in der Kombination gestrichen. Wir brechen die Suche ab, wenn wir die Endstellung gefunden haben oder die genannten Bedingungen nicht erfüllt sind. In einem typischen Backtracking-Verfahren können wir so alle interessanten Züge erreichen. Mit diesem Verfahren kann man erstaunlich schnell viele Spiele berechnen. Aber auch mit diesem Verfahren wird es sehr, sehr lange dauern, um alle Zugfolgen für eine Anfangs- und Zielkonstellation zu berechnen. Alternativ kann man aber bei einer bestimmten Konstellation sich aus den erlaubten Zügen einen Zug zufällig auswählen und diesen weiter verfolgen. Je nach Wahrscheinlichkeitsdichte für erfolgreiche Spiele, gelingt es in wenigen Sekunden bis mehreren Minuten mit einem modernen PC, einen Weg zu einem Spiel zu berechnen.

Ewald Rieger

Triceps spielt Solitaire

Ewald Rieger
ewald.rieger@t-online.de

Abstract

Im Artikel "Triceps - ein einfacher Roboter zur Demonstration von Pick und Place Aufgaben" (VD Ausg. 02 2003) wurde der Aufbau des Roboter kurz umrissen und die Grundfunktion zur Ansteuerung des Roboters beschrieben. Auf der Suche nach einer Anwendung kamen wir auf die Idee mit dem Roboter Pflock-Solitaire zu spielen. In diesem Artikel stellen wir die Erweiterungen zum Spielen von Solitaire vor und erklären die Programmergänzungen zum Spielen von Zugfolgen eines Pflock-Solitairespiels.

1. Das Spielfeld

Da unser Roboter nur eisenmetallische Teile tragen kann, können wir keine Holzpflöcke wie im klassischen englischen Pflock-Solitaire verwenden. Um eine robuste und sichere Handhabung zu garantieren, entschieden wir uns für Metallkugeln, die in runde Bohrungen auf einem Aluminium-Brett abgelegt werden. Eine dreieckige, an den Rändern nach unten abgekantete Platte ist in der Mitte mit den 33 Bohrungen des englischen Pflock-Solitaire-Spiels versehen. Entlang der Außenkanten sind nochmals 3 Reihen mit je 11 Löchern zur Ablage

der vom Spielfeld entfernten Kugeln vorgesehen. Nach einem Spiel kann der Roboter die Kugeln selbst den seitlichen Ablagen entnehmen und ein neues Spiel vorbereiten.

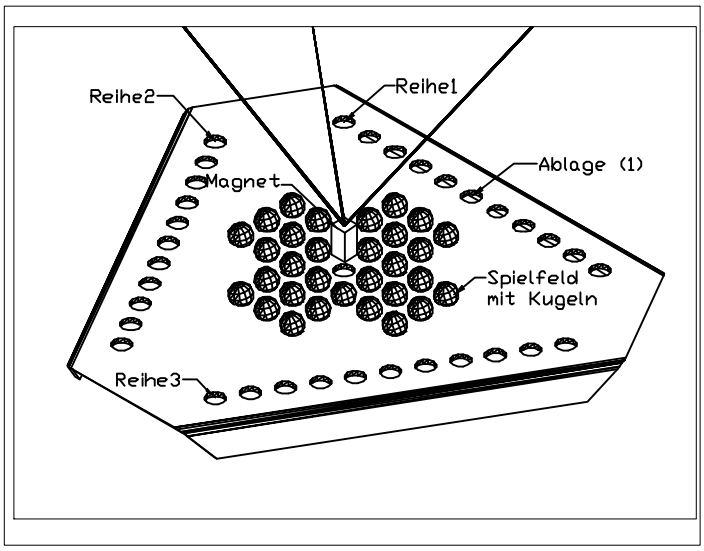


Abb.: Das Spielfeld

2. Programmierung

Das in diesem Artikel beschriebene Programm kann zwei unterschiedliche Befehlssätze erzeugen. Der erste Befehlssatz steuert den echten, aus Materie bestehenden Roboter an. Da aber nicht jeder das Geld und die Voraussetzungen zum Aufbau der Mechanik hat, entstand der Wunsch, nach einem virtuellen Gegenstück. Deshalb entstand ein zweiter Befehlssatz, der keine Mechanik ansteuert, sondern alle Zustandsinformationen des Systems für den virtuellen Roboter im Speicher ablegt. Reale mechanische Eigenschaften, z.B. die Elastizität der Seile unter Einfluß der Schwerkraft und Änderungen der Traglast beim Aufnehmen einer Kugel, zwangen uns, einige Routinen des Programms für die virtuelle Darstellung anders zu parametrieren. Grundsätzlich könnten beide Systeme auch parallel arbeiten. Dies gelingt aber nur zufriedenstellend, mit einem Mehrprozessorsystem auf dem der reale Roboter durch ein echtzeitfähiges System (z.B. einem Mikrokontroller) gesteuert und die Graphik auf dem zweiten System berechnet und dargestellt wird.

Der nachfolgende Text erklärt die Funktionen des Programms.

```
\ : simulation ;
```

Das Forth-Wort simulation ist eine Compiler-Direktive. Ist dieses Wort im Vokabular vorhanden, erzeugt der Compiler Code für den Simulator. Andernfalls erhalten wir den Code zur Ansteuerung des richtigen Roboters.

```
include circle4.fb
include triceps.fb
2 16 thru
```

Zunächst wird das Paket circle4.fb geladen, das Code zum Handhaben von Objekten in Listen erzeugt. Danach laden wir das File triceps.fb, um Code für den Roboter zu erzeugen.

```
[IFDEF] simulation
include trigraph.m solitare open [THEN]
```

Bei Bedarf compilieren wir zusätzlich den Simulator.

2.1 Die Geometrie des Spielfeldes

Zunächst benötigen wir einige Befehle, um den Greifer über den Löchern des Spielfeldes und der Ablage zu positionieren. Für alle Bewegungen in horizontaler Richtung wählen wir einen Abstand zum Brett, in der keine Kollision möglich ist. Die Vektorvariable *gr-org* enthält die Referenzposition des Greifers, die sich über der Mitte des Spielfeldes befindet.
Alle Berechnungen in den Programmteilen beziehen sich auf diese Ursprungsposition. Die Längen sind in 1/10 mm Einheiten angegeben.

```
: spiel-feld ( x-nr  y-nr -- )
  200 * swap 200 * swap 0 ( x y z )
  -600  -600 -200 v+
  gr-org pos@ v+ absbewegen ;
```

In Falle des Spielfeldes nimmt das Wort bei seinem Aufruf die Reihen- und Spalten-Nummern eines Loches auf dem Spielfeld vom Datenstack und berechnet daraus die absolute Position über dem Loch und positioniert den Greifer an diese Stelle.

```
: reihe1 ( nr -- )
  200 * 0 0 ( x y z )
  -1000 -1000 -200 v+
  gr-org pos@ v+ absbewegen ;

: reihe2 ( nr -- ) 200 * 750 + dup
   s>f pi f2* !6 f/ fsin f* f>s swap
   s>f pi f2* !6 f/ fcos f* f>s swap 0
  -1750 -1000 -200 v+
  gr-org pos@ v+ absbewegen ;

: reihe3 ( nr -- ) 200 * 750 + dup
   s>f pi f2* !6 f/ fsin f* f>s swap
   s>f pi f2* !6 f/ fcos f* f>s negate swap 0
  1750 -1000 -200 v+
  gr-org pos@ v+ absbewegen ;
```

Die Forth-Wörter *reihe1*, *reihe2*, und *reihe3* nehmen bei ihrem Aufruf die Lochnummer innerhalb einer dieser Reihen vom Datenstack und errechnen die absolute Position über diesem Loch und positionieren den Greifer an diese Stelle.

```
Create reihen ' reihe1 , ' reihe2 , ' reihe3 ,
DOES> swap cells + perform ;

: ablage ( nr -- )
  11 /mod reihen ;
```

Die drei Ablagereihen *reihe1, reihe2, reihe3* werden in dem Forth-Wort Reihen zusammen gefaßt. Bei seinem Aufruf nimmt das Wort den Index für eine Reihe vom Datenstack und ruft die entsprechende Reihe auf. Schließlich regelt das Wort *Ablage* den gemeinsamen Zugriff auf alle Kugelablagen.

```
Variable freie-ablage
33 freie-ablage !
```

Für die Verwaltung der freien Ablageplätze ist die Variable *freie-ablage* zuständig, sie zeigt immer auf den nächsten freien Platz in der Ablage.

2.2 Spielfeldverwaltung

```
create sol-field ( n m -- adr )
2 , 2 , 1 , 1 , 1 , 2 , 2 ,
2 , 2 , 1 , 1 , 1 , 2 , 2 ,
1 , 1 , 1 , 1 , 1 , 1 , 1 ,
1 , 1 , 1 , 0 , 1 , 1 , 1 ,
1 , 1 , 1 , 1 , 1 , 1 , 1 ,
2 , 2 , 1 , 1 , 1 , 2 , 2 ,
2 , 2 , 1 , 1 , 1 , 2 , 2 ,
DOES> -rot 7 cells * swap cells + + ;
```

Das Wort **sol-field** merkt sich den Spielbrettzustand in einer 7 * 7 Matrix . Nicht besetzbare Positionen sind durch eine 2 in der Matrix markiert. Eine 1 zeigt ein besetztes und eine 0 ein freies Loch auf dem Spielbrett an. Bei seinem Aufruf nimmt es den Spalten- und Reihenindex vom Datenstack und gibt die Adresse des entsprechenden Platzes zur weiteren Bearbeitung zurück.

```
Create #sol-field 49 cells allot

0 0 sol-field #sol-field 49 cells cmove

: init-sol-field #sol-field 0 0 sol-field 49 cells cmove ;

: sol-field-store ( n1 .. n49 -- )
   7 0 DO 7 0 DO I' I - 1- J' J - 1- sol-field !
      LOOP LOOP ;

: pos? ( n m -- flg ) \ gültige Position
  0 7 within >r 0 7 within r> and ;
```

Die Aufgabe von *pos?* ist die Reihen- und Spaltenindices auf ihre Gültigkeit zu prüfen. Beide Werte werden verbraucht und bei gültigen Werten durch ein True-Flag ersetzt.

```
: frei?    ( n m -- flg )
  sol-field @ 0= ;
```

Das Wort *frei?* prüft auf einen freien Platz auf dem Spielbrett. Bei seinem Aufruf nimmt es den Reihen- und Spaltenindex vom Datenstack und gibt bei einem freien und erlaubten Platz in der Spielfeldmatrix ein True-Flag zurück.

```
: ablegen  ( n m -- )
  2dup frei? 0= abort" nicht frei"
  2dup sol-field @ %1 or -rot sol-field ! ;
```

Ablegen belegt ein freies Loch auf dem Spielfeld. Bei seinem Aufruf nimmt es den Reihen- und Spaltenindex von Datenstack und setzt das zuständige Bit in der Matrix. Ist dies nicht möglich, wird mit der Meldung: "nicht frei" abgebrochen.

```
: belegt?  ( n m -- flg )
  sol-field @ 1 and ;
```

Ist der Platz an der Position n m auf dem Spielfeld belegt, erhalten wir ein True-Flag zurück.

```
: wegnehmen ( n m -- )
  2dup belegt? 0= abort" nicht belegt"
  2dup sol-field @ -2 and -rot sol-field ! ;
```

Wegnehmen entfernt eine Kugel aus dem Spielfeld. Dabei nimmt es den Spalten- und Reihenindex vom Datenstack und löscht das zuständige Bit in der Spielfeldmatrix. Ist keine Kugel vorhanden, wird mit der Fehlermeldung: "nicht belegt" abgebrochen.

```
: .sol-field
  cr space space 7 0 DO I 1 u.r LOOP
  7 0 DO cr I .
  7 0 DO
     I J belegt? IF 'X emit
     ELSE I J frei? IF 'O emit ELSE space THEN THEN
     LOOP LOOP cr ;
```

.sol-field druckt den Spielfeldzustand auf dem Terminal aus. Dabei steht das Zeichen X für einen belegten und das Zeichen O für einen freien Platz auf dem Spielfeld.

```
: summe  ( -- n ) 0
  7 0 DO 7 0 DO I J belegt? IF 1+ THEN LOOP LOOP ;
```

Das Wort *summe* zählt bei seinem Aufruf die Kugeln auf dem Spielfeld und gibt die Summe auf den Datenstack zurück.

2.3 Kugeln aufnehmen und ablegen

Zunächst definieren wir ein paar Forth-Worte zum Aufnehmen und Ablegen der Kugeln auf dem Spielfeld und der Ablage. Vor dem Ausführen dieser Befehle muß der Greifer bereits über der Kugel in der vereinbarten Höhe positioniert sein. Danach wird der Greifer um eine bestimmte Strecke relativ zu dieser Position abgesenkt, der Magnet eingeschalten und die Kugel auf die Ausgangsposition angehoben.

```
[IFDEF] simulation
: (kugel-ablegen   ( n m -- )
    300 down ablegen magnet off
    300 up ;
```

Der Roboter lernt spielen

```
: (kugel-aufnehmen ( n m -- )
    300 down wegnehmen magnet on
    300 up ;
```

Die Worte *(kugel-ablegen* und *(kugel-aufnehmen* werden auf das Spielfeld angewendet.

```
: ((kugel-ablegen
    300 down magnet off 1 freie-ablage +!
    300 up ;

: ((kugel-aufnehmen
    300 down magnet on -1 freie-ablage +!
    300 up ;
```

Während die Worte *((kugel-ablegen* und *((kugel-aufnehmen* bei der Ablage zur Anwendung kommen.

```
: kugel-wegnehmen ( n m -- )
  2dup spiel-feld (kugel-aufnehmen ;

: kugel-ablegen   ( n m -- )
  2dup spiel-feld (kugel-ablegen ;
```

Kugel-aufnehmen und *Kugel-ablegen* erwarten den Reihen- und Spaltenindex einer Kugel auf dem Datenstack und positionieren über diese Kugel. Danach wird die Kugel vom Greifer aufgenommen oder abgelegt.

```
: kugel-entfernen
  freie-ablage @ ablage ((kugel-ablegen ;

: kugel-holen
  freie-ablage @ 1- ablage ((kugel-aufnehmen ;
```

Kugel-entfernen und *Kugel-holen* kommen sinngemäß auf der Ablage zur Anwendung. Bei ihrem Aufruf wird die Position der Kugel der Variablen freie-ablage entnommen.

```
[THEN]
```

Nun kommen wir zur Definition, der für den realen Roboter notwendigen Befehle.

```
[IFUNDEF] simulation

: (kugel-aufnehmen  magnet bitclr 250 down 250 up ;

: (kugel-ablegen   250 down magnet bitset 250 up ;

: kugel-wegnehmen ( n m -- )
  spiel-feld (kugel-aufnehmen ;

: kugel-ablegen   ( n m -- )
  spiel-feld (kugel-ablegen ;

: kugel-entfernen
  freie-ablage @ ablage (kugel-ablegen 1 freie-ablage +! ;

: kugel-holen
  1 negate freiablage +! freiablage @ ablage
  (kugel-aufnehmen ;

[THEN]

: vernichten ;
```

Die unterschiedlichen Definitionen für den Greifer waren notwendig, weil beim richtigen Roboter der Magnet schon zu Beginn des Greifens eingeschaltet wird, damit er bei Absenken nicht an der Kugel vorbei pendelt. Auf der anderen Seite muß dem Simulator genau zum richtigen Zeitpunkt über die Variable freie-ablage das Berühren der Kugel mitgeteilt werden.

2.4 Züge überprüfen

Die nachfolgenden Worte führen einen kompletten Zug in eine bestimmte Richtung aus und verändern dabei den Zustand der Spielfeld-Matrix. Sie wirken aber nicht auf den Roboter oder den Simulator. Ihre Anwendung liegt darin, eine Zugfolge vor dem eigentlichen Spiel auf Durchführbarkeit zu überprüfen.

```
: ziehen1 ( n m -- )
   2dup         wegnehmen
   2dup >r 2- r> ablegen
      >r 1- r> wegnehmen vernichten ;
```

Ziehen1 erwartet an der Position n m eine Kugel und zieht nach Westen,

```
: ziehen2 ( n m -- )
  2dup wegnehmen 2dup 2+ ablegen
  1+ wegnehmen vernichten ;
```

während *ziehen2* seinen Zug nach Osten ausführt.

```
: ziehen3 ( n m -- )
  2dup         wegnehmen
  2dup >r 2+ r> ablegen
  >r 1+ r> wegnehmen vernichten ;
```

Ziehen3 macht einen Zug nach Süden

```
: ziehen4 ( n m -- )
  2dup wegnehmen 2dup 2- ablegen
  1- wegnehmen vernichten ;
```

Und zum Schluß noch *ziehen4*, das einen Zug nach Norden ausführt.

```
Create ziehe ( n m flg -- )
' 2drop   ,
' ziehen1 ,
' ziehen2 ,
' ziehen3 ,
' ziehen4 ,
DOES> swap cells + perform ;
```

Das Forth-Wort *Ziehe* faßt nun die Züge in die vier Einzelrichtungen zusammen. Bei seinem Aufruf erwartet das Wort den Reihen- und Spaltenindex einer Kugel, gefolgt vom Richtungsflag und führt den Zug mit dieser Kugel in die gewünschte Richtung aus.

2.5 Roboter spielt Solitaire

Die Forth-Worte zum Spielen arbeiten in gleicher Weise, wie die Worte im Abschnitt: "Züge überprüfen". Nur, daß sie noch zusätzlich die Befehle zum Roboter weiterleiten und damit richtig spielen. Deshalb verzichte ich auf die Erklärung dieser Worte.

```
: spiele1 ( n m -- )
   2dup          kugel-wegnehmen
   2dup >r 2- r> kugel-ablegen
      >r 1- r> kugel-wegnehmen kugel-entfernen ;

: spiele2 ( n m -- )
   2dup kugel-wegnehmen 2dup 2+ kugel-ablegen 1+
   kugel-wegnehmen kugel-entfernen ;

: spiele3 ( n m -- )
   2dup          kugel-wegnehmen
   2dup >r 2+ r> kugel-ablegen
      >r 1+ r> kugel-wegnehmen kugel-entfernen ;

: spiele4 ( n m -- )
   2dup kugel-wegnehmen 2dup 2- kugel-ablegen 1-
   kugel-wegnehmen kugel-entfernen ;

Create spiele ( n m flg -- )
' 2drop   ,
' spiele1 ,
' spiele2 ,
' spiele3 ,
' spiele4 ,
DOES> swap cells + perform ;
```

2.6 Ein Beispiel für das Zentralumkehrspiel

Das klassische Pflock-Solitaire-Spiel besteht darin, das Spielfeld mit allen Pflöcken zu belegen, nur der Platz in der Mitte des Feldes bleibt frei. Von dieser Ausgangstellung beginnen wir möglichst viele Pflöcke zu entfernen. Gewonnen hat derjenige, bei dem am Schluß des Spiels genau der Pflock auf dem mittleren Platz des Spielfeldes übrig bleibt. Der nachfolgende Text zeigt eine mögliche Zugfolge dieses Spiels.

```
( zug: 3 5 4 zug: 1 4 3
  zug: 3 3 2 zug: 3 1 2
  zug: 1 2 3 zug: 2 0 2
  zug: 5 4 1 zug: 3 2 1
  zug: 0 2 3 zug: 2 3 4
  zug: 2 6 4 zug: 3 4 1
  zug: 0 3 3 zug: 0 4 3
  zug: 3 6 4 zug: 4 0 1
  zug: 3 4 1 zug: 4 6 4
  zug: 2 0 2 zug: 2 2 2
  zug: 1 4 3 zug: 3 4 3
  zug: 4 2 2 zug: 6 3 1
  zug: 3 3 3 zug: 6 2 1
  zug: 4 1 2 zug: 4 3 2
  zug: 6 4 1 zug: 4 5 4
  zug: 5 3 1 ) ok
```

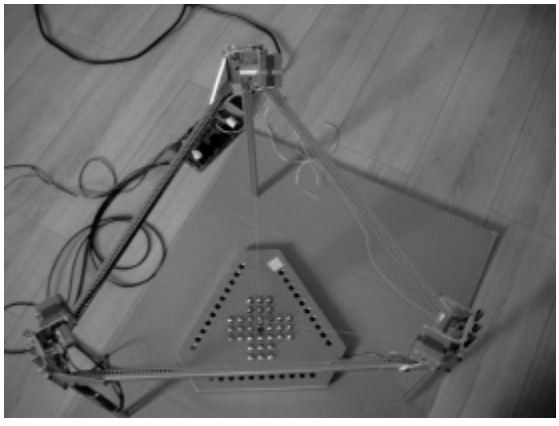
Der nachfolgende Programmtext führt ein komplettes Spiel auf dem Roboter aus. Zunächst bestücken wir das Spielfeld manuell mit den Kugeln und referenzieren das System. Das Objekt *Anfang* ist ein Spielfeld-Objekt aus dem, mit der Methode *fetch*, die Spielfeld-Matrix auf den Daten-Stack gelesen wird. *Sol-field-store* schreibt nun diese Werte in das Spielfeld des Roboters. Durch die Befehlsfolge *freie-ablage off* teilen wir dem System mit, daß die Ablage für aus dem Spiel zu entfernende Kugeln leer ist. *Ausgefuehrte* ist ein Objekt der Klasse FIFO, in dem die Züge als Objekte-Liste von der Klasse *Zug:* gespeichert sind. Nun übergeben wir die Listenlänge als Parameter an die interpretative DO-LOOP-Schleife.
Innerhalb der Schleife holen wir bei jedem Schleifendurchlauf einen Pointer auf den aktuellen Zug und lesen mit der Methode *zug>* die Parameter auf den Datenstack. Das Wort *spiele* nimmt diese Werte vom Stack und führt den Zug aus. Nach 31 Zügen ist das Spiel zu Ende.

```
anfang fetch sol-field-store freie-ablage off

ausgefuehrte len dup 0

[DO] dup [I] 1+ - ausgefuehrte tes [with] zug> [endwith]
spiele [LOOP] drop
```

3. Ausblick

Das vorgestellte Programm zeigt, wie man den Roboter zum Solitaire-spielen bringt. Allerdings ist es ziemlich mühevoll sich neue Spiele auszudenken. In einem weiteren Artikel stellen wir ein Programm vor, das in der Lage ist, jedes erdenkliche Solitairespiel in wenigen Minuten zu berechnen. Damit ist der Spaß beim Spielen garantiert.

Ewald Rieger



Stellen Sie sich eine Programmiersprache vor, interaktiv wie Basic, schnell wie Assembler, strukturiert wie Pascal und beliebig erweiterbar. Eine solche Sprache ist Forth. Mit diesen Worten startet Bernd Stejskal seine Web-Präsenz zu Forth. Recht hat er, natürlich. Vielleicht besuchen Sie ihn einfach einmal im Netz.

anton@mips.complang.tuwien.ac.at (Anton Ertl)

|Die **EuroForth 2003** findet vom 17.-19. Oktober in Ross-on-Wye (in der |Naehe von Birmingham) statt. Naehere Informationen gibt's auf |http://www.micross.co.uk/euroforth2003/Call_for_papers.html.

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 39, August 2003

Datenstrukturen mit CREATE DOES> bauen
Ben Koehorst

Alles über CREATE ... DOES> . Für CHForth und 8052-ANS-Forth. Datenstrukturen mit CREATE ablegen, am Beispiel von VARIABLE erklärt. "Für Datenworte, die mehr können, als nur die Adresse ihres Datenfeldes auf den Stack zu legen, gibt es das Wort DOES>" . Erklärung und allgemeine Formulierung.

Die Website der holländischen Forth-Gruppe:

http://www.forth.hccnet.nl/nieuws

Beispiel eines Forth-Treffens (9.8.2003)

10.30 Saal geöffnet für Kaffee und Frühankommer
11.00 Datenstrukturen mit CREATE DOES> bauen (Ben Koehorst)
12.30 Pause und Gelegenheit zur informellen Kontaktpflege
13.30 CREATE DOES> in ByteForth (Willem Ouwerkerk)
14.45 Schluss

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

Arbeitsgruppe MARC4
Rafael Deliano
Tel./Fax: (**0 89) - 8 41 83 17** (p)

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!

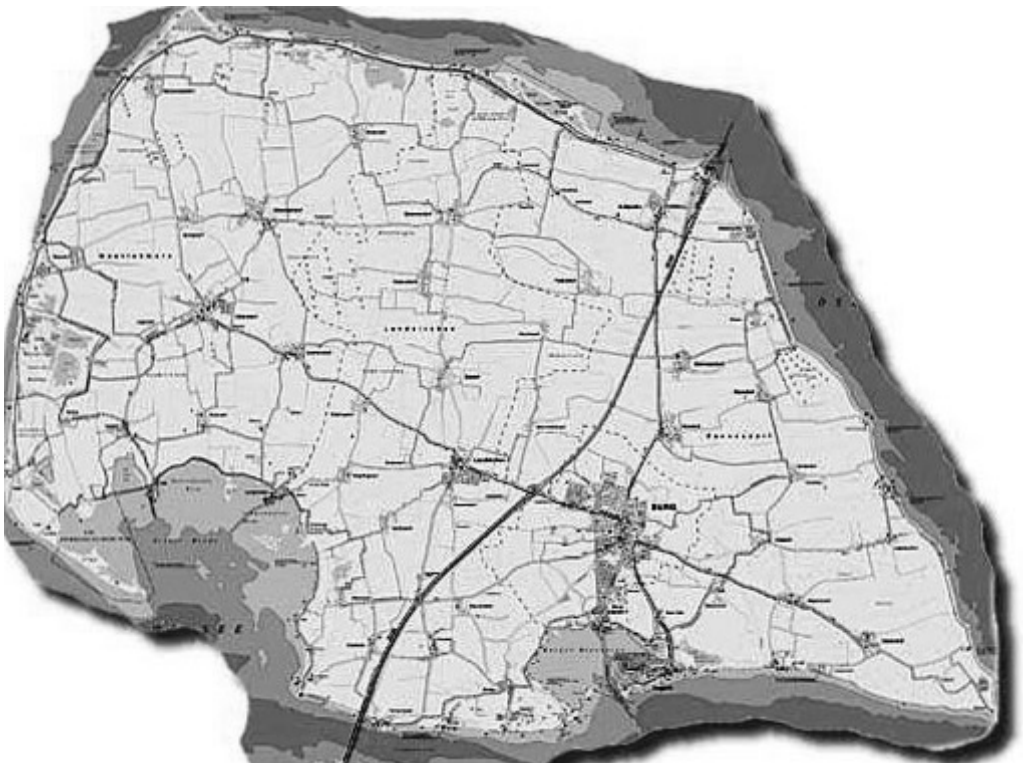
Hinweise zu den Angaben nach den Telefonnummern:

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

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

3 / 2003

Fehmarn, der sonnenreichste „Flecken“ Deutschlands



Auf die Ostseeinsel Fehmarn lädt Ulrich Hoffmann die Forthgemeinde ein, zur

Jahrestagung
und
Mitgliederversammlung
2004

Dort wird, in der Internetpräsenz der Insel nachzulesen, folgendes angeboten:

Windsurfen	(sicher nicht im April)
Siloclimben	(clever, die Friesen)
Segeln	(vielleicht kommt jemand mit dem eigenen Boot?)
Fahrradfahren	(es gibt sicher auch Taxis)
Kart-Sport	(Ja, wenn die sich mit Forth steuern lassen)
Beachvolleyball	(siehe Windsurfen)
Tauchen	(nur für Eisbären)
Golf	(die Löcher muß man selbst mitbringen)
Angeln	(das freut den sportlichen Redakteur)
Tennis	(...muß man ja nicht)
Reiten	(nur, wenn an die Pferde nichts dran kommt)
Kitesurfing	(was immer das auch ist)
Hochseilgarten	(siehe Tennis)
Kutterfahren	(ist schöner als Angeln, wenn die See ruhig ist)
Rundflüge	(kommt jemand mit dem eigenen Flugzeug?)
Inline-Skating	(siehe Tennis)

Forthtagung	(steht (noch) nicht in der Internetpräsenz)

Neben der Forthtagung bietet Fehmarn reichlich Kurzweil, auch für Ihre Begleitung. Haben Sie den Ausflug schon eingeplant?