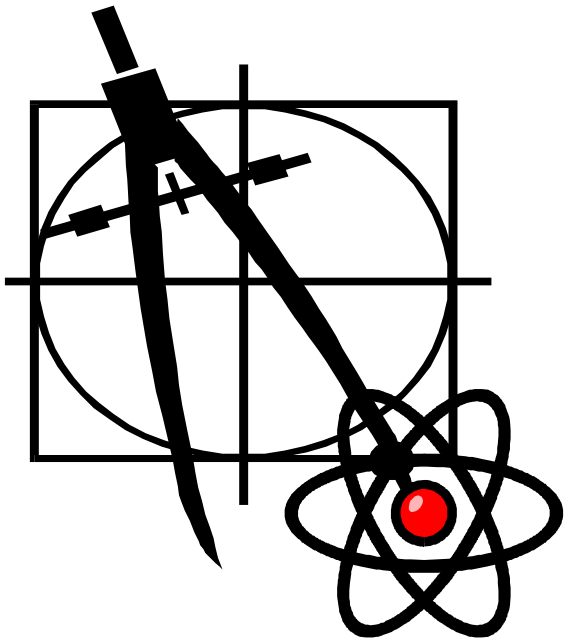
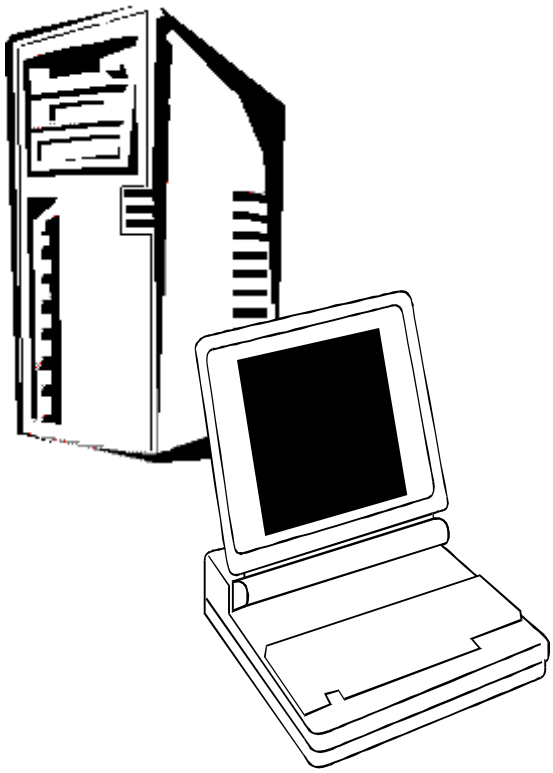
# VIERTE DIMENSION

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





## In dieser Ausgabe:




15. Jahrgang - **Ausg. 2/'99** - 10,- DM

Organ der FORTH-Gesellschaft e.V.

# Dienstleistungen und Produkte fördernder Mitglieder des Vereins

# Editorial


**IMPRESSUM**

**Name der Zeitschrift**

***Vierte Dimension***

**Herausgeberin**

Forth-Gesellschaft e.V.
Postfach 161204
D-18025 Rostock
Tel.: 0381-4007828
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./Fax: 02841-58 3 98
E-Mail:

VD@FORTH-EV.DE
FRIEDERICH.PRINZ@T-ONLINE.DE

**Anzeigenverwaltung**

Büro der Herausgeberin

**Redaktionsschluß 1999**

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

**Erscheinungsweise**

1 Ausgabe / Quartal

**Einzelpreis**

DM 10,- zzgl. Porto u. Verp.




Liebe Leser,

Es ist CEBIT-Zeit. Die Nachrichtentechnik feiert sich selbst, wie in jedem Jahr, auf jeder CEBIT. Längst schon meint digitale Technik nicht ‚den Computer schlechthin', sondern das gesamte, riesige Spektrum der Informations– und Komunikations-technik. Und riesig ist dieses Spektrum wirklich – so riesig, daß es erstmals in der BRD am Gesamtumsatz aller Branchen einen größeren Anteil für sich verbuchen konnte als die Automobilindustrie. Das will schon etwas heißen. ‚Des Deutschen liebstes Kind' auf den zweiten Platz zu verdrängen, hat man bisher keiner Branche zugetraut !

Da wundert es nicht, daß Mancher diesen Vergleich der Größenordnungen anzweifelt. Immerhin werden auf der CEBIT alle Produkte des Metiers zusammengezählt, von der Branchen-Softwarelösung für Schneider-Fachbetriebe, über Handys und andere Spieleconsolen bis zu ‚fertigen' Netzwerkumgebungen aus > 5.000 PC nebst Anlagen der mittleren Datentechnik. Die Automobilhersteller zählen Fahrräder, Inlineskater und Snowboards nicht zu ihrem Branchenumsatz.

Und auch wenn sich die ‚Handymanie' noch längst nicht ausgetobt hat – wie sich allerorten durch bloßes Hinschauen feststellen läßt – gibt es bereits erste Anzeichen dafür zu vermelden, daß die eine oder andere Modewelle über kurz oder lang an Kraft verlieren und zusammenbrechen wird. Immerhin durfte ich erst kürzlich erleben, daß ein Zeitgenosse sein Handy verschämt in der Tasche seines Sakkos verschwinden ließ, als ich – natürlich scherzhaft gemeint – in einer Gesprächsrunde mit ernster Mine versicherte, kein Handy zu benötigen, weil ich meinem Arbeitgeber so wichtig sei, daß der mich zu finden wisse, wenn ich im Betrieb benötigt würde.

Ob die ‚digitale Zunft' auf dem Platz Eins der Umsatzriesen bleiben kann, muß abgewartet werden. Vielleicht wächst die Zahl derjenigen, die sich von den 72 neuen MMX Befehlen und 9,5 Mio Transistoren des Pentium III ebenso wenig beeindrucken lassen wie von dem neuen 3DNow des AMD K6 III – und ganz gemächlich auf den Pentium VI warten, weil der III-er dann für ‚einen Apfel und ein Ei' zu haben sein wird. Was in absehbarer Zukunft sicher bleiben wird, ist eine vielleicht weniger stürmisch, aber kontinuierlich wachsende Branche, sind > 70.000 Informatiker, die jedes Jahr zu wenig ausgebildet werden und ist der Bedarf an Menschen, die noch wirklich wissen, was sie tun – wenn sie etwas mit einem digitalen Gerät tun. Was wachsen wird, ist die Anzahl theoretischer Konzepte, die in der Praxis nicht funktionieren, weil einfache Selbstverständlichkeiten übersehen werden. Hier möchte ich nicht auf Details eingehen, sondern lediglich auf die überall wachsende Zahl von Upgrades, Service Packs, Fixes, Help-Tools und Flash-Releases verweisen, die letztlich ‚einen Grund haben' müssen.

Was damit auch wachsen wird, ist vielleicht nicht der Bedarf an Forth, aber sicher der Bedarf an Forthern ! Forther suchen für einfache Probleme nach einfachen Lösungen. Forther beschreiben komplexe Probleme mit einer Summe von einfachen Lösungen. Forther arbeiten an jedem Problem in angemessener Weise ! Und das tun WIR sozusagen in einer für uns ganz natürlichen Umgebung – eben mit Forth.

Ich wünsche der ‚digitalen Branche', daß sie doppelt so umsatzstark werden wird, wie es die Automobilbranche einschließlich Fahrradherstellern und Inlineskates ist.

*fep*

**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*

# Leserbriefe

Betreff: Microcontrollerverleih

Der Microcontrollerverleih der FG erwartet Ihre Anfrage. Bitte wenden Sie sich an:

Thomas Prinz
Adalbert-Stifter-Str. 2
69412 Eberbach

Email : topeban@forth-ev.de
micro@forth-ev.de

---

Betreff: E-Mail J. Merkel
Von: Fred Behringer

Lieber Fritz,

endlich mal jemand, der mir sagt, daß ich nicht ins Leere hinein arbeite. Das gibt mir Mut für vieles mehr. Ich hänge Dir statt vieler Worte einfach mal die gesamte E-Mail von Joachim Merkel, über die ich mich sehr freue, an.

Herzliche Grüße

Fred

===== E-Mail von Joachim Merkel an mich ==========

Lieber Herr Behringer,

ich freue mich über Ihre Kurzinfos aus Forth-Veröffentlichungen anderer Forth- und Sprachgemeinschaften. Deren Auswertung und redaktionelle Überarbeitung führt für mich zu einer erheblichen geistigen Gewinnsteigerung bei der Lektüre der VD. Kurz gesagt, ich finde Ihren Einsatz für Forth außerordentlich vielseitig und anregend.

Ich bitte Sie um die Überlassung des gesamten Artikels "Turbo-Forth, Conways "Life" und die DNF als If-Bedingung". Ich hatte von einem anderen Forth-Gesellschaft-Mitglied die Quellen eines life-Programms für ein Z80-Forth bekommen, (da ich mich 'noch' gerne mit Z80-Programmierung befasse) mit der Auflage es auf ein (80x86) PC-Forth zu übertragen. Dazu kommt mir Ihr Beitrag wie gerufen.

Das Turbo-Forth auf Deutsch würde ich gerne mal in Augenschein nehmen. Sie hatten die dt. Übersetzung des Help-Files gemacht, habe ich in Erinnerung.

Herzlichen Dank
und freundlichen Grüße,

Joachim Merkel

*Liebe Leser,*

*ALLE unsere Autoren arbeiten ‚just for fun' für uns alle. Sie tun dies in ihrer Freizeit und selbstverständlich unentgeldlich. Die Vorbereitung eines Artikels nimmt gelegentlich mehr Zeit in Anspruch, als es die Aufarbeitung des jeweiligen Themas getan hat. Und in der Regel macht das Schreiben des Artikels auch weniger Spaß als das jeweilige Thema.*
*Trotzdem finden sich immer wieder Forther bereit, uns allen das aufzubereiten, womit sie sich gerade auseinander gesetzt haben. Das ist ein großes Lob wert, das ich an dieser Stelle im Namen der Forthgesellschaft allen unseren Autoren aussprechen möchte.*
*Darüber hinaus wäre es für einen Autor nicht nur angenehm, eine Rückmeldung zu einem Artikel zu bekommen, sondern bei zukünftigen Arbeiten sicher auch eine große Hilfe. In diesem Sinne bitte ich um Ihre konstruktive Kritik. Schreiben Sie uns – oder dem jeweiligen Autoren direkt – wenn Ihnen etwas mißfallen hat, wenn Sie ein Thema gerne aus einem anderen Blickwinkel betrachtet sehen würden oder wenn Sie ein Thema in der VD vermissen. Bitte schreiben Sie auch, wenn Ihnen etwas gefallen hat. Machen Sie den Autoren Mut, mit ihrer Arbeit fortzufahren.*

*fep*

Hallo Fritz,

seit der Tagung bei Euch habe ich mich ziemlich in Holon vergraben und tauche erst jetzt langsam wieder auf und strekke wieder die Fühler aus...
Wie geht es Dir?
Die VD funktioniert ja bestens auch in der "billigen" Art, und Eure gute Arbeit produziert eine immer wieder sehr willkommene Zeitschrift.
Es gibt eine neue Holon Version, eine neue Website und ein neues Konzept der Verbreitung. Ich konzentriere mich nun voll auf das Web, nachdem ich endlich kapiert habe, dass es zum Marketing auf herkömmliche Art mehr als einen alten Physiker braucht. Somit werde ich auch nicht mehr an der Embedded Systems Messe die Forth Flagge zeigen, was mir sehr leid tut, aber der Klaus Flesch macht das ja noch bestens ...

Für heute kann ich Dir nur die folgende Eigenwerbung als Beitrag zur VD anbieten.

**HolonForth** ist überarbeitet worden und bietet in der neuen **Version 4.3** ein noch besseres Handling. Die ungewöhnlichen Vorteile dieser voll integrierten Forth Cross-Entwicklungsumgebung bleiben erhalten. U.a.:

- Browser Benutzerschnittstelle (von Smalltalk inspiriert).
- Strukturierte Darstellung des Quellcodes.
- Hypertext und direkter Zugriff auf jedes Programmwort.
- Automatischer Codeaustausch bei Redefinition eines Wortes.
- Automatisches Codestripping.
- Einzelschritt Debugging von Forth und Assembler Code.

HolonForth läuft unter DOS und ist verfügbar als:

***weiter auf Seite – 6 –***

# Inhalt



In der nächsten Ausgabe finden Sie voraussichtlich:

„Wieviel Windows braucht der Mensch ?" - Früchte zählen mit ‚klassischem' OO-Forth

Das Original eines Aufsatzes der bereits in der Forthwrite Nr. 100 erschienen ist
Friederich Prinz

Hashing – wie geht das eigentlich ?

Jeder weiß was es ist, die ‚reine Lehre' praktiziert (aus gutem Grund) Niemand und jedes Forth macht es anders

# Leserbriefe

Hinweis in eigener Sache

In den Ausgaben 3/98 und 4/98 der VD war versehentlich im Impressum für das **Forthbüro** in Rostock eine falsche Telefonnummer angegeben. Die korrekte Rufnummer lautet :

**0381-4007828**

Unter dieser Nummer ist das Forthbüro sowohl telefonisch erreichbar, als auch per FAX.

Die Redaktion der VD ist ab sofort ebenfalls telefonisch und per FAX erreichbar, unter der Rufnummer:

**02841-58398**

Holon 86 für x86 Systeme (ideal für embedded PCs). Erzeugt real mode Code für DOS Programme. Der gesamte DOS Speicherbereich ist für Programmcode verfügbar. Holon 86 entwickelt interaktiv in einen separaten (embedded) PC (Nabelschnur Methode). Alternativ können Sie interaktiv zwischen zwei DOS Fenstern unter Windows (oder einem anderen Multitasking Betriebssystem) entwickeln. Jede Programmänderung ist sofort im Zielsystem wirksam. Eine FREIE Testversion kann unter ***http://holonforth.com/tools/holon86.htm*** bezogen werden.
Holon 11 für 68HC11 Mikrokontroller. Verwendet einen 200 Bytes Monitor, der im Boot Mode des 68HC11 geladen wird. Sie können direkt entwickeln an jedem Zielsystem, das über die serielle Schnittstelle SCI ansprechbar ist und das in den Bootmode versetzt werden kann. Weitere Hardware ist nicht nötig. Eine Testversion von Holon 11 mit voller Funktion (erzeugt EPROM-fähigen Code) ist FREI verfügbar unter ***http://holonforth.com/tools/holon11.htm***.

Ich hoffe ja, gelegentlich wieder nützlichere Beiträge zu liefern, aber jetzt möchte ich endlich Holon ins Rollen bringen. Vielleicht gibt es ja auch mal Leute, wie Du, die die Ideen aufgreifen und weiterführen, und dann könnte es wertvolle Diskussionen geben auch in der VD. Tatsache ist, dass das Konzept in der Praxis bestens funktioniert - und noch weiter ausgebaut werden könnte.
Für heute beste Grüsse, *Wolf*

## Ein LOGO für die Forthgesellschaft

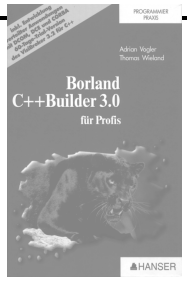
Sie erinnern sich an den Aufruf in der letzten VD ? Ein neues neues LOGO braucht die Gesellschaft. Dieses LOGO soll die FG zukünftig auf Briefpapier, auf unseren WEB Seiten und überall dort vertreten, wo ein hoher Wiedererkennungswert,gefragt' ist. SIE, die Leser der VD und Mitglieder der FG sollten dieses LOGO erstellen. Es ist IHR LOGO !

Eines haben wir bereits hier !

Aber die FG hat sicher mehr ‚kreative' Mitglieder. Darum wiederholen wir hier den Aufruf, entsprechende Vorschläge einzureichen. ...und nicht vergessen: es gibt einen interessanten Preis zu gewinnen !

*fep*

*Adrian Vogeler, Thomas Wieland*




## Borland C++ Builder 3.0 für Profis

Der Borland C++ Builder überspannt sehr viele Programmieraufgaben: Einerseits gestattet er, die alten C++Programme in eine moderne Form zu überführen (32-Bit-Anwendung, Industriestandard C++, RAD für Windows 95/98, NT). Andererseits enthalten seine Bibliotheken mächtige Komponenten zur Erstellung von mehrfach benutzten Objekten, Datenbankanwendungen und mehrschichtigen Applikationen. Mit dem C++Builder ist also eine einheitliche Behandlung vieler, ganz unterschiedlicher Anwendungen möglich. Man arbeitet auf der bereits bekannten Delphi-Oberfläche. Zur Umstellung alter Programme befindet sich auf der Distribution das Programm IdeToBpr. Schließlich enthält die CD des Rezensenten auch noch das Lehrbuch: 'C++ Builder in 14 Tagen'. Dieses hilft, Anfänger an die Arbeit heran zu führen.

Mit ihrem Buch und der CD ***Borland C++ Builder 3.0 für Profis*** richten sich Vogeler und Wieland nur an wirklich erfahrene Programmierer. Der behandelte Stoff beginnt erst dort, wo die Aufgaben nicht mehr mit dem 'klassischen standalone Programm' abzuhandeln sind. -- Hier einige Stichworte aus dem Inhaltsverzeichnis:

> Botschaften des Betriebssystems -- Komponentenentwicklung -- Die Objektmodelle COM und CORBA -- DLLs sind nicht behandelt -- OLE server und clients -- ActiveX-Controls und ActiveX-Formulare -- OLE-Container -- Datenbankgrundlagen -- Client/ ServerProgrammie-rung -- Applikationen mit MIDAS -- DCE-basierte Multi-tier-Anwendungen -- CORBA-basierte Multi-tier-Anwendungen -- Ein großes Kapitel mit Tips und Tricks.

Jedes Kapitel hat eine ausführliche, leicht verständliche Einführung in den kommenden Stoff. Danach geht es zur Sache, und der Computer muß eingeschaltet sein. An Hand von Lesestoff und ausgetestetem Beispiel dringt der Studierende in die modernen Programmiertechniken vor. -- 'Borland C++ Builder 3.0 für Profis' ist dafür ein sehr gutes Hilfsmittel. -- Vielleicht hätte man ja zukünftige Leser dadurch ermutigen können, daß man dem Buch außer dem Index auch noch ein Verzeichnis der zahlreichen Abkürzungen beigefügt hätte. Dieses Verzeichnis würde sicher auch später von den Arbeitenden öfters benutzt. Sogar, während der Computer eingeschaltet ist.

*Ulrich Richter*

Carl Hanser Verlag
ISBN 3-446-19549-1
568 Seiten, Hardcover
DM 79,-- / ÖS 577,-- / SFr 72,--

# OO-Forth, klein und fein

von
Chris Jakeman
50 Grimshaw Road
Peterborough
PE1 4ET
c.jakeman@bigfoot.com

Schon Viele haben sich an objektorientierten Erweiterungen von Forth versucht, einschließlich Pountain, Losh, Ertl, Zimmer, McKewan und sogar Charles Moore selbst. Einige dieser Erweiterungen sind Zusatzpakete für ANS Forth geworden, andere sind in die jeweiligen Forth-Systeme integriert, um maximale Performance zu gewährleisten, die meisten sind riesig.

In einer Umgebung, in der gilt: „small is beautiful“, verdient das was Bernd Paysan hierzu anbietet, eine eingehendere Betrachtung. Sein minimales OOF erweitert ein beliebiges ANS Forth um einfache Vererbung und Polymorphismus, und verpackt Methoden und Daten gemeinsam in einem Objekt – so wie objektorientierte Pakete es tun sollten. Late-Binding ist als ‚Verfahren‘ voreingestellt, aber das Early-Binding kann – zur Erhöhung der Ausführungsgeschwindigkeit – im Bedarfsfall erzwungen werden. Welche andere Programmiersprache

```
: METHOD ( m v -- m' v ) CREATE  OVER , SWAP CELL+ SWAP
       DOES> ( ... o -- ... ) @ OVER @ + @ EXECUTE ;
: VAR ( m v size -- m v' ) CREATE  OVER , +
       DOES> ( o -- addr ) @ + ;
: CLASS ( class -- class methods vars ) DUP 2@ ;
: END-CLASS  ( class methods vars -- )
  CREATE  HERE >R , DUP , 0 ?DO ['] NOOP , LOOP
  CELL+ DUP CELL+ SWAP @ 2 CELLS - R> 2 CELLS + SWAP MOVE ;
: DEFINES ( xt class -- ) ' >BODY @ + ! ;
: NEW ( class -- o )  HERE SWAP DUP , @ 1 CELLS - ALLOT ;
: :: ( class "name" -- ) ' >BODY @ + @ COMPILE, ;
CREATE OBJECT  1 CELLS , 2 CELLS ,
```

kann diese Fülle an Funktionalität mit 12 Zeilen definieren ? Einige Dinge kann dieses ‚Mini-OOF‘ nicht bieten. Alle Namen bleiben sichtbar. Informationen können nicht ‚versteckt‘ werden. Die Syntax könnte ordentlicher sein und kürzer, und zur Compilezeit finden keine Überprüfungen statt. Aber die hier eingeführte Syntax ist für kleine Applikationen akzeptabel und in jedem Fall gut genug für Experimente.

Ein näherer Blick

Um den Code leichter verfolgen zu können, sollten wir uns ein Beispiel ansehen, und anschließend prüfen, wie der oben gezeigte Code arbeitet. Hierzu werden wir eine Klasse BUTTON definieren, die dazu genutzt werden soll, Text auf dem Bildschirm auszugeben. Ganz der objektorientierten Art entsprechend, sollen die Daten jedes Buttons, sowie die Zeiger auf dessen Methoden zur Intialisierung und zur Ausgabe, in ein und der selben Struktur zusammengefaßt werden, in einem Objekt. Die Klasse BUTTON erbt von der Klasse POSITION, welche Bildschirm-Koordinaten für eine Reihe von Objekten – einschließlich BUTTONS – enthält.
Zum Beispiel:

```
OBJECT CLASS  \Die neue Klasse
              \erbt von der
              \voreingestell-
              \ten Klasse
              \Objekt
CELL VAR X    \Jedes POSITION
              \Objekt enthält
              \Koordinaten
              \über seine
CELL VAR Y    \Position auf
              \dem Bildschirm
END-CLASS POSITION \ Erzeuge
              \ den Klassen-
              \namen POSITION.
POSITION CLASS \Die BUTTON Klasse erbt
              \von POSITION X & Y
CELL VAR TEXT  \Jedes BUTTON Objekt
              \enthält einen Zeiger auf
              \seinen Text
CELL VAR LEN   \sowie die Angabe über
              \dessen Länge
METHOD INIT
\Die BUTTON Klasse kennt
\eine Methode zur Initia-
\lisierung und Erzeugung
\ von Objekten,
METHOD DRAW
\sowie eine weitere zur
\ Erledigung der Bild
\ schirmausgaben
END-CLASS BUTTON
\END-CLASS vervollständigt
\ die neue Klasse.
```

Zu diesem Zeitpunkt sind die Namen der beiden Methoden INIT und DRAW bekannt, aber nicht deren Funktionen. Das kommt gleich.

Wir haben jetzt drei Klassen: OBJECT, POSITION und BUTTON. Das sind CREATEd Worte, die, wenn sie ausgeführt werden, die Adresse ihrer Datenstrukturen auf dem Stack hinterlegen:

```
OBJECT -->    &+0   1  cell  Größe eines Objektes der
                             Klasse Objekt
              &+1   2  cells Größe der Klasse Objekt
POSITION --> &+0    3  cells Größe eines Objektes der
                             Klasse Position
              &+1   2  cells Größe der Klasse
                             Position
BUTTON -->    &+0   5  cells Größe eines Objektes der
                             Klasse Button
              &+1   4  cells Größe der Klasse Button
              &+2   ' NOOP   Execution Token der
                             voreingestellten Methode
              &+3   ' NOOP   Execution Token der
                             voreingestellten Methode
```

# OO-Forth, klein und fein

Die Datenstrukturen der Klassen enthalten die beiden Größen, und die Execution Token ihrer Methoden. Wenn die Klassen erzeugt werden, werden die zugehörigen Daten in den Speicherplätzen abgelegt, die ihnen durch VAR zugewiesen wurden, sowie im ersten Feld einen Zeiger auf diese Klassendaten.

Um die Klasse BUTTON zu vervollständigen, müssen wir noch zwei Methoden definieren. Jede der beiden Methoden ist ein anonymes Wort, erzeugt mit dem ANS :NONAME, das dann in die Methode eingebunden wird, die mit DEFINES exakt benannt ist.
Die DRAW Methode nutzt AT-XY, um den Cursor zu positionieren und TYPE, um den Text des BUTTON Objektes auf dem Bildschirm auszugeben.

```
:NONAME ( &Object -- ) \ Hier wird die gleiche
                       \ Syntax wie beim @ genutzt,
                       \ Methoden erwarten die
  >R                   \ Adressen ihrer Objekte auf
                       \ Stack.
  R@ X @ R@ Y @ AT-XY  \ VARs X und Y geben einen
                       \ Offset in die Daten des
                       \ Objekts zurück
  R@ TEXT @ R> LEN @ TYPE \ Ebenso verhalten sich
                       \ die VARs TEXT und LEN.
; BUTTON DEFINES DRAW  \ DEFINES nutzt ' um DRAW
                       \ zu finden, und bindet
                       \ alles zusammen.
```

Die INIT Methode setzt alle Daten in einem BUTTON Objekt:

```
:NONAME ( x y addr u &Object -- )
  >R  R@ LEN !  R@ TEXT !  R@ Y !  R> X !
; BUTTON DEFINES INIT
```

Einer der großen Vorteile aller objektorientierten Ansätze ist der Polymorphismus. Das ‚Verhalten' von Objekten und Klassen ist veränderbar. In unserem Beispiel benötigen wir eine weitere Klasse von BUTTONs, die wir BOLD-BUTTON nennen wollen, und die ihren Text in irgend einer Weise hervorgehoben ausgeben soll. Selbstverständlich wird diese neue Klasse von BUTTON erben. Wir könnten die Hervorhebung der Textausgaben dadurch erzwingen, daß wir die Methode DRAW neu definieren. Aber das Selector Wort :: ermöglicht es uns, auch die Methode DRAW wieder zu verwenden. Das Beispiel:

```
BUTTON CLASS \ Die neue Klasse erbt von der Klasse
             \ BUTTON
END-CLASS
BOLD-BUTTON  \ Zusätzliche VARiablen oder Methoden
             \ werden nicht benötigt

: BOLD ( -- )  [CHAR] * EMIT
;

:NONAME ( &Object -- ) >R
  R@ X @ 1-  R@ Y @ AT-XY
  R> BOLD [ BUTTON :: DRAW ] BOLD
; BOLD-BUTTON DEFINES DRAW
```

Ebenso wie DEFINES sucht das Wort :: in der Klasse BUTTON nach DRAW und kompiliert den gerade definierten Code mit einem Link von der Methode zu dem Code. (:: kann ebenso dazu benutzt werden, das Early-Bindung zu erzwingen, was effizienter wäre, aber gegen die Regeln des Polymorphismus verstoßen würde.)
Nun sind die Klassen vollständig. Wir können Objekte erzeugen. Wir nutzen NEW, um Speicher für ein anonymes Objekt zu allokieren und CONSTANT, um den zugehörigen Namen zu erzeugen, der bei seinem Aufruf die Adresse des Objektes auf dem Stack hinterläßt.

```
BUTTON NEW CONSTANT BUTTON1
BOLD-BUTTON NEW CONSTANT BOLD-BUTTON1
```

Sobald die Objekte erzeugt sind, können wir Ihnen die gewünschten Daten zuweisen:

```
11 5 S" plain message" BUTTON1 INIT
10 6 S" bold message" BOLD-BUTTON1 INIT
```

Jetzt haben wir zwei Objekte; BUTTON1 und BOLD-BUTTON1. Das sind CREATEd Worte, die bei ihrem Aufruf die Adresse ihrer Datenstrukturen auf dem Stack hinterlegen.

```
BUTTON1 -->      &+0  Adresse  Zeiger auf die
                                Klasse BUTTON
                 &+1  11       x Koordinate
                 &+2  5        y Koordinate
                 &+3  Adresse  Zeiger auf den
                                String "plain
                                message"
                 &+4  13       Länge des
                                Strings

BOLD-BUTTON1 --> &+0  Adresse  Zeiger auf die
                                Klasse
                                BOLD-BUTTON
                  &+1  10      x Koordinate
                  &+2  6       y Koordinate
                  &+3  Adresse Zeiger auf den
                                String "bold
                                message"
                  &+4  12      Länge des
                                Strings
```

Schließlich lassen wir die beiden Objekte mit den folgenden Anweisungen arbeiten:

```
PAGE
BUTTON1 DRAW      \ Nutzt das DRAW in BUTTON
BOLD-BUTTON1 DRAW \ Nutzt das DRAW in
                  \ BOLD-BUTTON
```

***Chris Jakeman*** *ist Mitglied der* ***FIG UK*** *und verantwortlicher Editor der Forthwrite.*
*Der hier abgedruckte Artikel ist in der Forthwrite Nr.: 99, Nov. ´98 unter dem Titel ‚****Object-Oriented Forth – A Minimal Approach****' erschienen.*
*Übersetzung und Abdruck erfolgen mit der freundlichen Genehmigung des Autors.*
*Die Übersetzung wurde von Friederich Prinz vorgenommen.*
*Red.*

Das erzeugt die Bildschirmausgabe:

```
 plain message
*bold message*
```

Und wie funktioniert das Ganze ?

Die Auskommentierung von Bernd Paysans 12-Zeiler sollte diese Frage klären helfen:

```
1 CELLS CONSTANT CELL            \ Eine Abkürzung

: CLASS                          \ Vor dem Aufbau einer neuen Klasse den Zugriff auf die Da-
ten
                                 \ des Erblassers geben (Parent)
       ( &ParentClass -- &ParentClass ClassSize ObjectSize )
       DUP 2@                    \ Eine Klasse ohne Methoden enthält nur zwei
                                 \ Größenangaben, die eine steht für ein Objekt, die
                                 \ zweite für eine abgeleitete Klasse
;

: METHOD ( ClassSize ObjectSize -- NewClassSize ObjectSize )
                                 \ Jede Methode macht die Klasse ein wenig größer
       CREATE                    \ Der Methode einen Namen geben
       OVER ,                    \ Die aktuelle Klassengröße sichern, sie entspricht dem
                                 \ Offset auf die Methode innerhalb dieser Klasse.
                                 \ Ein Zeiger auf den ausführbaren Code wird später
                                 \ von DEFINES hier abgelegt.
       SWAP CELL+ SWAP           \ Die Klassengröße erhöhen, damit Platz für neue
                                 \ Methoden bleibt
     ( ... &Object -- ... )
     DOES> ( -- &Object &MethodData )
       @                         \ Hole den Offset auf die Daten der Methoden
       OVER @                    \ Das erste Feld im Objekt zeigt auf die Daten der
                                 \ Klasse
       + @ EXECUTE               \ Ausführen des xt (CFA), der in der Klasse ab dem
                                 \ Offset der Methode hinterlegt ist
;

: VAR ( ClassSize ObjectSize VarSize --
        -- ClassSize NewObjectSize ) \ Jede Variable vergrößert ein Objekt
          CREATE                   \ Der Variablen einen Namen geben
       OVER ,                      \ Die neue Größe des Objektes sichern (s. Method)
       +
     ( ... &Object -- ... )
        DOES> ( &Object &Offset -- &Data )
          @ +
;

: END-CLASS  ( &ParentClass ClassSize ObjectSize -- )
       CREATE                              \ Der neuen Klasse einen Namen geben
       , DUP ,                             \ Objekt und Klassengröße kompilieren
                  \ -- &ParentClass ClassSize )
       HERE >R                             \ Adresse der ersten Methode
       2 CELLS ?DO  ['] NOOP ,  CELL +LOOP \ Für jede Methode NOOPs kompilieren
       CELL+                               \ Get Adresse der Größe von Parent
       DUP CELL+                           \ und Adresse der Methoden von Parent
       SWAP @       \ -- &ParentMethods ParentSize )
       2 CELLS -                           \ abzüglich der beiden Größenfelder
       R> SWAP MOVE                        \ Kopiere die Methoden von Parent
;                                          \ in die neue Klasse

: DEFINES ( xt &Class "method-name" -- )
       ' >BODY @              \ Hole den Offset der nachfolgenden Methode innerhalb
                              \ der Klasse, Ersetze den xt von NOOP durch die
       + ! ;                  \ CFA der echten Methode

: NEW ( &Class -- &Object )
       HERE >R                             \ Adresse des neuen Objektes sichern
       DUP @ ALLOT                         \ benötigten Speicher allokieren
       R@ !                                \ Adresse der Klasse im ersten Feld des
       R> ;                                \ Objektes sichern
```

```
: :: ( &Class "method-name" -- )
       ' >BODY @
\ DEFINES sichert das xt und :: liest
\ es auf dem gleichen
       + @           ( -- xt )
\ Weg und kompiliert es. :: darf nur
\ innerhalb einer
       COMPILE, ;
\ Definition verwendet werden.
```

Zum Schluß wird eine leere Basisklasse erzeugt, von der nachfolgende Klassen erben können:

```
CREATE OBJECT  1 CELLS , \Platz für
                         \leere Objekte
               2 CELLS , \und Klassen-
                         \größe
```

Zusammenfassung

Warum sich um Objekte kümmern ? Alles in allem können sie rein gar nichts für Sie erledigen, was Sie nicht auch anders regeln könnten. Ein Schlüsselvorteil ist aber sicher eine sauberere Organisation Ihrer Definitionen, besonders dann, wenn Sie eine Sammlung von Einheiten verwalten müssen, die sich jeweils nur gering von einander unterscheiden, so wie BUTTON und BOLD-BUTTON. Die Methoden der Objekte helfen dabei, ähnliche Dinge auch zusammen zu halten. Das macht Verzweigungsstrukturen überflüssig, die sonst eine beliebige Applikation kräftig ‚aufblähen' können. Das Beste an Objekten ist aber die Möglichkeit, völlig neue Einheiten zu definieren, wie z.B. COLOUR-BUTTON, und dabei den bereits getesteten Code einfach weiter zu verwenden.

Die 12 Zeilen Code enthalten im Grunde alles was Sie brauchen, um Ihre Applikationen objektorientiert anzulegen. Es gibt keinen Grund mehr, das nicht wenigstens zu versuchen.

*Chris Jakeman*



# Strings in Forth

Eine Betrachtung und Erweiterung der
zeichenverarbeitenden Wörter
für Forth-83 und ANS-Systeme


von
Nils Eilers
eilers@ocean.com


String-Packages gibt es wohl, seit es FORTH gibt. Warum dennoch nur wenige Wörter Eingang in die Standards 83 und ANS finden konnten, bleibt rätselhaft. Ernsthafte Stringverarbeitung kann man mit den Worten des optionalen string-word-sets jedenfalls kaum betreiben.

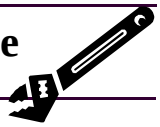
Der Forther hilft sich mit trickreichen Konstruktionen à la "VARIABLE Name 30 ALLOT" oder "Name COUNT TYPE". Eingaben erledigt er mit "Name 2 + 30 EXPECT SPAN @ Name !".

Wer nun aber echte Stringverarbeitung braucht, der kann bei Forth das Fürchten lernen. Um so unverständlicher erscheint dem Autor, daß sich bei der Suche nach zuvor erstellten String-Packages nichts rechtes finden ließ.

Das nun vorliegende String-Package soll einen Hauch von Komfort ermöglichen, wie man ihn aus anderen Sprachen gewohnt ist. Der Autor hat sich hierbei bewusst nicht an C oder seinem Derivat Java orientiert, obgleich dies von manchen Forthern für den Fall, daß Forth Lücken lässt, gefordert wird. Vielmehr diente das altbewährte und unausrottbare BASIC als Vorlage.
Dabei sind die String-Wörter der Sprache C dem Autor durchaus bekannt, doch ein lesbares und somit wartbares Programm stellt er sich anders vor.

## Grundlagen

Man unterscheidet bei der Verwaltung von Zeichenvariablen solche mit Count (wie z.B. aus Pascal und BASIC bekannt) von sog. Null-terminierten Zeichenketten (wie z.B. bei C). Der Unterschied liegt - wie der mit seiner Maschine sicher vertraute Forther sicher weiß - darin, wie die Länge des Strings und somit seine Begrenzung gehandhabt wird.



Strings mit Count legen in einem dem String in der Regel vorgestellten Byte oder Wort die Länge ab. Nullterminierte Strings hingegen enden lediglich mit einem ASCII NULL. Beide Varianten haben so ihre Vor- und Nachteile.

Wird bei den Counted Strings lediglich ein Byte für die Längenangabe verwendet, beschränkt sich die maximale Länge auf dementsprechend 255 Bytes. Für viele Fälle ist dies einfach zu wenig.
Die nullterminierten Strings hingegen dürfen selbst kein ASCII NULL enthalten, da dies ja das Ende des Strings markieren würde. Dies scheint auf den ersten Blick keine größere Einschränkung zu sein, doch bringt das den Nachteil mit, daß man sich z.B. bei der Längenberechnung des Strings oder bei anderen Zugriffen auf das Ende des Strings nicht auf einen festen Wert berufen kann, sondern sich jedes Byte "einzeln" ansehen muß. Dies soll an einem kurzem technischem Beispiel verdeutlicht werden: von dem Inhalt einer Stringvariablen sollen lediglich die letzten vier Zeichen verwandt werden. Hierzu muß die Gesamtanzahl der im String enthaltenen Zeichen, seine Länge, bekannt sein. Bei einem counted string stellt dies kein Problem dar: die Maschine ruft lediglich kurzerhand den count-Wert ab.
Anders sieht es bei null-terminierten Strings aus: da die Länge nirgendwo gespeichert wird, muß sie jedesmal zu Fuß bestimmt werden: die Maschine muß sich jedes Byte einzeln ansehen und mit ASCII NULL vergleichen, um auf diese Art und Weise schließlich die Länge bzw. die Adresse des letzten Bytes zu ermitteln. Wie der Autor leider feststellen musste, kann der Overhead in einem massiv stringverarbeitenden Programm gewaltig sein.

Deshalb wurde hier versucht, den Mittelweg zu gehen. Strings werden hier mit einem 16-bit-Count gehandhabt, was also Strings von einer Länge bis zu 65535 Zeichen erlaubt. Dies sollte für über 99% der Anwendungen ausreichend sein.
Das Package wurde vollständig in "Hochsprachen-Forth" programmiert; es findet sich also nicht eine einzige CODE-Definition darin. Sicherlich ist damit eine Menge Overhead verbunden, doch das ist der Preis der Portabilität.

## Interne Verwaltung

Strings werden generell in der Form c-addr len übergeben. C-addr stellt hierbei einen Zeiger auf das erste Zeichen des Strings dar, len seine Länge.
Zu beachten ist hierbei, daß viele String-Wörter nicht den Speicherinhalt direkt bearbeiten, sondern lediglich die Parameter c-addr len. Dies stellt keinerlei Beeinträchtigung dar, solange das Ergebnis der Stringoperation weiterverarbeitet werden soll. Soll das Ergebnis jedoch in der Quellvariablen gespeichert werden, sind also Ursprung und Ziel der Stringmanipulation identisch, muß das Ergebnis der Manipulation ausdrücklich mittels $COPY wieder zugewiesen werden.

Intern werden Strings wie andere Wörter auch in's Wörterbuch kompiliert. Dies geschieht u.a. mittels CREATE. Ihr weiterer Aufbau ist wie folgt:

```
                60          s
Aufsteigende   /^\   59     n
Adressen        |    58     a
                |    57     H   Erstes Stringzeichen
                |    56       0 \Aktuelle Länge LEN
                55   4       / ( hier: Intel's Little Indian )
                54   0       \Maximale Länge MAXLEN
                53   20        /  ( hier: Intel's Little Indian )
```

Die folgenden Wörter werden intern dazu benutzt, sich innerhalb des Headers zu bewegen. Wer das String-Package erweitern oder modifizieren möchte, wird von diesen Wörtern Gebrauch machen:

```
: >LEN    2 - ;          ( c-addr -- count  )
: >MAXLEN 4 - ;          ( c-addr -- maxlen )
: LEN>    2 + ;          ( count  -- c-addr )
: MAXLEN> 4 + ;          ( maxlen -- c-addr )
```

## Definition und Zuweisung von String-Variablen

Der einfachste Fall besteht darin, eine String-Konstante zur weiteren Verwendung mittels S" ( -- adr len \ " ) vorzubereiten. S" ( -- adr len \ " ) stellt den wohl einfachsten Fall dar, eine Stringkonstante vorzubereiten. S" folgt ein Leerzeichen zur Tokentrennung, dann der konstante Wert des Strings und schließlich ein Anführungszeichen (") um das Ende zu markieren. Als Ergebnis werden auf dem Stack c-addr len übergeben; wobei c-addr die Adresse des ersten Textzeichens darstellt und len die Anzahl der enthaltenen Zeichen. Dieses Wort arbeitet ausschließlich im interpretativen Modus.

C" ( -- adr len \ " ) entspricht S" bis auf die Tatsache, daß es ausschließlich im compile-mode, also während der Definition neuer Worte verwandt werden kann. Diese beiden Wörter haben sich als "quasi-Standard" etabliert und sind in diversen Implementationen umgesetzt.

$CONSTANT ( adr len -- ) . weist einer benannten Stringkonstanten einen zuvor mittels S" oder C" vorbereiteten Wert zu. Der Aufruf erfolgt üblicherweise in der Form s" IBM" $CONSTANT BigBlue. Entgegen der Erwartung darf der Inhalt einer "Konstanten" noch beliebig verändert werden: dies kann nicht bemerkt, geschweige denn verhindert werden. Abhilfe würde ein weiterer Eintrag im Header des Strings schaffen, der aber auch vor jeder Stringoperation interpretiert werden müsste, was das Programm unnötig aufblähen und die Ausführungsgeschwindigkeit nachhaltig negativ beeinflussen würde. Lediglich die Länge wird automatisch bestimmt und zur maximalen Länge dieser de-facto-Variablen erklärt.

Variablen werden vorab für eine bestimmte Maximallänge definiert. Dieser Platz wird für sie im Wörterbuch statisch vorgesehen und kann weder freigegeben, noch vergrößert

oder verkleinert werden. $VARIABLE ( maxlen -- \ ) legt eine neue Variable an, die mit einem leeren String der Länge Null initialisiert wird und maximal "maxlen" viele Zeichen fassen kann. Beispiel: 30 $VARIABLE Vorname .

**$DEL** ( addr len -- ) eignet sich hervorragend dazu, einer Stringvariablen einen definierten Anfangswert zuzuweisen: nämlich einen Leerstring. Der Inhalt der Variablen wird also gelöscht.

**$+** ( s-addr s-len d-addr d-len ) hängt den Quellstring s an den Zielstring d an.
Beispiel: Vorname Name $+ .

**$COPY** ( s-addr s-len d-addr d-len -- ) kopiert den Inhalt der Variablen s nach dem Inhalt der Variablen d, der dabei natürlich überschrieben wird. Aber Vorsicht: es wird nicht geprüft, ob der Zielstring den Quellstring überhaupt fassen kann!

## Zeichenketten und Schnittchen

Hier finden wir sie wieder, die altbekannten Wörter LEFT, RIGHT und MID; diesmal in der Form:

```
$LEFT    ( c-addr1 len1 sublen -- c-addr2 len2 )
$RIGHT   ( c-addr1 len1 sublen -- c-addr2 len2 )
$MID     ( c-addr1 len1 position sublen -- c-addr2
           len2 )
```

**$LEFT** liefert die ersten sublen Zeichen von links an, oder eben so viel, wie der String zu bieten hat.

**$RIGHT** macht analoges von rechts.

**$MID** liefert einen sublen Zeichen langen Ausschnitt gerechnet ab der Position " position ". Hierbei ist zu beachten, daß das erste Zeichen des Strings nicht wie in C das nullte, sondern ganz analog zu BASIC das 1. Zeichen ist.

```
S" FORTH" 2 3
$MID TYPE liefert also "ORT" und nicht etwa "RTH".
```

**$SUB** ( c-addr1 len1 first last -- c-addr2 len2 ) möchte ich den Forthern besonders an's Herz legen, da sich mit diesem Wort alle drei oben genannten Wörter vortrefflich ersetzen lassen. Es liefert einen Ausschnitt aus dem String c-addr1 len1, beginnend mit dem Zeichen Nr. first und endend mit dem Zeichen Nr. last

Beispiel:

```
S" And the Lord brought us FORTH"
$CONSTANT s
s 1 3          $SUB TYPE   ==>   "And".
s dup 5 swap   $SUB TYPE   ==>   "FORTH".
s 15 5         $SUB TYPE   ==>   "rough".
```

**$CUT** ( c-addr1 len1 c-addr2 len2 pos len1' -- ) schneidet aus dem String c-addr2 len2 einen Teilstring aus. Anders als bei $MID wird er aber nicht nur nach c-addr1 len1 übergeben, sondern wird tatsächlich aus dem Quellstring entfernt.
Beispiel: Wort Zeile 4 10 $CUT

**$INS** ( c-addr1 len1 c-addr2 len2 pos -- ) fügt den String c-addr2 len2 ab Position posin den String c-addr1 len1 ein.

Beispiel: Hierhin s" Laß mich rein!" 4 $INS

## Ein- und Ausgaben

Eingaben

Hierzu kennt das Standard-FORTH das Wort EXPECT ( c-addr len -- ) . Es erwartet (expect: englisch für "erwarte") vom Benutzer eine Eingabe über die Tastatur, die mit ENTER abgeschlossen werden muß. Dabei können maximal len Zeichen entgegengenommen werden. Die eingegebenen Zeichen - und nur diese! - werden dann beginnend mit der Adresse c-addr abgelegt.

Wieviele Zeichen nun denn tatsächlich eingegeben wurden, muß der Forther noch der User-Variablen SPAN entnehmen. Bis zu einem ordentlichen counted-string war es also ein langer Weg. Nehmen Sie die Abkürzung:

Eingaben erledigt von nun an das Wort **$INPUT** ( c-addr len -- ) . Die Länge len bezeichnet hierbei allerdings im Gegensatz zu EXPECT nicht die maximal einzugebenden Zeichen, sondern die Länge des Strings vor der Eingabe, die verworfen wird. Sie wird lediglich mitübergeben, um einfache Aufrufe in der Form Name $INPUT zu ermöglichen. Unter Name wird dabei die Benutzereingabe als counted-string abgelegt.

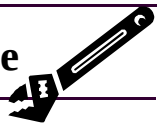
Anschließend wird direkt mitabgespeichert, wieviele Zeichen eingegeben wurden. Das interne Wort MAXLEN ( c-addr len -- maxlen ) liefert $INPUT die Information, wieviele Zeichen der String maximal fassen kann, so daß nichts "daneben" geht.

Ausgaben

Ausgaben erledigt nach wie vor das Wort TYPE , das von diesem Package nicht angerührt wird. An der einfachen Benutzung Bezeichner TYPE gibt es nichts mehr zu verbessern.

Bündige Ausgaben erreicht man mit den folgenden drei Wörtern, wobei die ersten beiden zu den F-PC-Bordmitteln zählen.
n1 bezeichnet hierbei die Breite des Feldes.

```
$.R   ( addr len n1 -- )    Rechtsbündige Ausgabe
$.L   ( addr len n1 -- )    Linksbündige Ausgabe
$.M   ( addr len n1 -- )    Zentrierte Ausgabe
```



## Vergleiche

Forth bietet von Haus aus leistungsfähige Wörter zum Vergleichen von Strings. Leider haben sie sich im Lauf der Jahre ein
wenig verändert, so daß der jeweilige Forther immer erst klären muß, nach welchem Standard sich seine Entwicklungsumgebung richtet: Forth-83 oder ANS.

Die drei Gebrüder COMPARE

Beide kennen das Wort COMPARE , doch benutzen sie es unterschiedlich. Forth-83 verwendet drei Stackparameter und vergleicht nur Strings gleicher Länge, bei ANS dürfen die Strings unterschiedlich lang sein. Ihr Ergebnis liefern beide in gleicher Form: als Flag f, das die Werte 0, -1 und +1 annehmen kann.
Die Aufrufe erfolgen bei Forth-83 in der Form COMPARE ( c-addr1 c-addr2 len -- f ), bei ANS aufgrund der möglicherweise unterschiedlichen Längen COMPARE ( c-addr1 len1 c-addr2 len2 -- f ). Ist das Flag 0, sind die Strings identisch. Ist das Flag negativ, wird der zweite String alphabetisch nach dem ersten einsortiert; ist er positiv, davor. Bei Strings unterschiedlicher Länge ist ein Sonderfall möglich: der längere String kann zu Beginn den kürzeren String enthalten. Hier gilt nun die Regel, daß der längere String alphabetisch danach eingeordnet wird.
Was hier in der Theorie furchtbar kompliziert klingt, ist in der Praxis ganz einfach: die Firma "Abflußreinigung Mayer" wird
im Telefonbuch vor der Firma "Abflußreinigung Mayer und Sohn" geführt.

Was das Unterscheiden von Groß- und Kleinbuchstaben gilt, so richtet sich COMPARE nach dem Wert der Systemvariablen CAPS. Ist CAPS logisch wahr, werden Groß- und Kleinbuchstaben nicht unterschieden; ist es hingegen logisch falsch, werden sie unterschieden. Zu beachten ist hierbei, daß ein amerikanisch orientiertes FORTH von den in diesem Sprachraum üblichen Umlauten nichts weiß und dementsprechend ein "Ä" und ein "ä" immer zweierlei sind.

COMP hingegen unterscheidet bei gleicher Parameterübergabe stets zwischen großen und kleinen Buchstaben, CAPS-COMP
hingegen nie.

## In FORTH heißt Babuschka $INSTR

$INSTR ( c-addr len sub-addr sublen first -- p ) prüft, ob im String "c-addr1 len1" der String "sub-addr sublen" enthalten ist. Dabei wird der String c-addr erst ab dem Zeichen Nr. first durchsucht; eventuelle frühere Vorkommen des Teilstrings in der zu durchsuchenden Zeichenkette bleiben unberücksichtigt. Es antwortet mit dem Wert p, der die Position des evtl. gefundenen Substrings angibt. Konnte der Substring hingegen nicht gefunden werden, ist p=0.

Beispiel:
s" Vierte Dimension" s" mens" 1 $INSTR . ==> Pos. 10.
s" Vierte Dimension" s" mens" 13 $INSTR . ==> NULL.

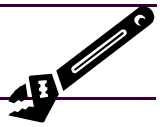
Der Beitrag von Nils Eilers liegt im Original als HTML-File vor. Dieses File, sowie die nachfolgenden Quellen, können per E-Mail von der Redaktion der VD ‚bezogen' werden.

*-red-*

```
\ STRING.INC   "Body" des String-Package, 16/32bit,
\ Nils Eilers, 1998/99
\ Letzte Änderung: 03.02.1999

comment:
  - wird von STRING.SEQ (16-bit, F-PC) oder
    STRING.F (32-bit, Win32For) geladen
  - Anmerkungen, Hinweise und Fehlermeldungen
    bitte an: eilers@ocean.gun.de
\ ------------------------------------------------
comment;

\ Befehle, um sich innerhalb des String-Headers zu
\ bewegen
: >LEN 2 - ;                    ( c-adr -- count )
: >MAXLEN 4 - ;                 ( c-adr -- maxlen )
: LEN> 2 + ;                    ( count -- c-adr )
: MAXLEN> 4 + ;                 ( maxlen -- c-adr )

: $.M   ( addr len n1 -- )
   TUCK UMIN TUCK - 2/ SPACES TYPE ;

: $CONSTANT                     ( adr len -- )
  CREATE DUP DUP (,) (,)        ( Maxlen und Laenge
  \                               sichern )
  DUP >R HERE SWAP MOVE
  R> ALLOT
  DOES> LEN> DUP (@) SWAP LEN> SWAP ;

: $VARIABLE             ( maxlen -- \ )
  CREATE DUP (,) 0 (,) ( Header creieren,
                         maxlen und len compilieren)
  ALLOT                 ( Platz fuer Zeichenkette
  \                       reservieren )
  DOES> DUP MAXLEN>     ( runtime: -- c-adr )
  SWAP LEN> (@) ;
                        (             len   )

: $DEL                  ( adr len -- )
  DROP >LEN 0 SWAP (!) ; ( String reinitialisieren:
  \                        Laenge auf 0 )

: $LEFT MIN ;           ( adr len sublen -- adr len )
                        ( Liefert die ersten sublen
    \                     Stueck Zeichen des Strings)

: $RIGHT                ( adr len sublen -- adr len )
  2DUP MIN
  >R                    ( save sublen )
  DROP R@ - +  R> ;     ( d-adr = s-adr + s-len -
    \                     sublen + 1 cell)
                        ( d-adr = Zieladresse:
    \                     destination )
                        ( s-adr = Quelladresse:
    \                     source )

: $MID                  ( adr len pos sublen
    \                     -- adr len )
  >R NIP + 1 - R> ;

: $SUB                  ( addr1 len1 first last
    \                     -- addr2 len2 )
                        ( first: 1..len1; last:
    \                     1..len2, last>=first )
```

```
  OVER - 1 +  ROT DROP
  -ROT + 1 - SWAP ;

: $LEN NIP ;           ( adr len -- len )

: $COPY                ( s-addr s-len d-addr d-len
    \                    -- )
  DROP >LEN 2DUP (!)
  LEN> SWAP MOVE ;

: $INPUT               ( adr len -- )
  DROP                 ( drop dummy-length )
  DUP >R               ( save c-adr )
  DUP >MAXLEN (@) EXPECT  ( expect up to [maxlen]
    \                       characters )
  SPAN (@) R> >LEN (!) ;  ( save length )
: MAXLEN               ( adr len -- maxlen )
  DROP                 ( drop dummy-length )
  >MAXLEN (@) ;

comment:
\ In F-PC sind UPC und UPPER bereits definiert!
: UPC                          ( char -- CHAR )
  DUP 97 < OVER 122 > OR IF EXIT THEN
  223 AND ;

: UPPER                        ( adr len -- )
  0 DO DUP I + DUP C@ UPC SWAP C! LOOP DROP ;
comment;

: LOC                          ( CHAR -- char )
  DUP 92 > OVER 65 < OR IF EXIT THEN
  32 XOR ;

: LOWER                        ( adr len )
  0 DO DUP I + DUP C@ LOC SWAP C! LOOP DROP ;

: $+      ( s-adr s-len d-adr d-len )
          ( haengt String s an String d hinten an )
  ROT 2DUP +                   ( Neue Laenge )
  >R ROT DUP >LEN R> SWAP (!)  ( Neue Laenge
     \                           speichern )
  ROT + SWAP MOVE ;            ( String kopieren )

: $>C                          ( adr len -- c )
  IF C@ ELSE 0 THEN ;

: C>$                         ( c -- adr len )
  1 PAD (!)  1 PAD LEN> (!)   ( Header aufbauen,
     \                          benutzt PAD )
  PAD MAXLEN> TUCK C! 1 ;     ( Zeichen speichern )


\ Wenn das Zielsystem lokale Variablen unterstützt,
\ darf's auch ruhig ein bißchen "moderner" zugehen.
\ Hier die "unverwüstliche" Variante:

VARIABLE c-addr1
VARIABLE c-addr2
VARIABLE len1
VARIABLE len2
VARIABLE len3
VARIABLE pos

: $INSTR ( c-addr len subaddr sublen pos -- p )
  pos (!)  len2 (!)  c-addr2 (!)  len1 (!)
  c-addr1 (!)
  len1 (@) len2 (@) - pos (@) DO
    len2 (@) 0 DO
      c-addr1 (@) J + I + C@
      c-addr2 (@) I + C@ =
      IF J pos (!)
      ELSE 0 pos (!) LEAVE
      THEN
    LOOP
    pos (@) IF LEAVE THEN
  LOOP
  pos (@) 1 +
;
```

```
: $CUT ( c-addr1 len1 c-addr2 len2 pos len3 -- )
  len3 (!)  pos (!)  len2 (!)  c-addr2 (!)  len1 (!)
  c-addr1 (!)
  ( Ausschnitt zuweisen )
  c-addr2 (@) len2 (@) pos (@) len3 (@) $MID
  c-addr1 (@) len1 (@) $COPY
  ( Länge verkleinern )
  len2 (@) len3 (@) - c-addr2 (@) >LEN (!)
  ( Rechtes "Ende" nach links schieben )
  c-addr2 (@) pos (@) + DUP len3 (@) + 1 - SWAP 1 -
  len2 (@) pos (@) - MOVE
;

: $INS ( c-addr1 len1 c-addr2 len2 pos -- )
  pos (!)  len2 (!)  c-addr2 (!)  len1 (!)
  c-addr1 (!)
  ( Länge vergrößern )
  len1 @ len2 @ + c-addr1 @ >LEN (!)
  ( Rechtes Ende verschieben: Platz schaffen )
  c-addr1 @ pos @ 1 - + DUP len2 @ + len1 @ pos @
  1 - -  MOVE
  ( String einfügen )
  c-addr2 @ ( Quelle ) c-addr1 @ pos @ + 1 -
  ( Ziel ) len2 @ MOVE
;

: .RULERCHAR ( n -- )
  DUP 10 MOD IF
    DUP 5 MOD IF
      ( insert TAB-processing code here! TOS=pos )
      DROP
      ASCII . EMIT
      ( . )
    ELSE DROP ASCII : EMIT THEN
  ( : )
  ELSE
    10 / BEGIN DUP 100 > WHILE 100 - REPEAT
    ASCII 0 + EMIT
  ( n )
  THEN
;

: RULER ( last+1 first -- )
  \ zeichnet ein Lineal, z.B.   71 0 RULER
  DO I .RULERCHAR LOOP
;

: $DUMP ( c-addr len -- )
  BASE @ >R
  CR 2DUP            ." c-addr: "
  SWAP 8 U.R         ."    len: "      5 .R
  CR OVER DUP >LEN @ ." [LEN]: "       5 .R
             >MAXLEN @ ."    [MAXLEN]: " 5 .R CR
  DUP 0= IF
    ." empty string!" CR
  ELSE
    0 DO
      DUP I DUP 5 .R + DUP 8 U.R C@ DUP 5 .R
      DUP HEX 5 .R DECIMAL 5 SPACES EMIT CR
    LOOP
    DROP
  THEN    R> BASE (!)
;

CR .( String-Unterstuetzung geladen. Nils Eilers,
      1998/99 -- freeware )

\ STRING.SEQ String-Package von Nils Eilers,
\ 1998/99, benötigt STRING.INC

: S" ASCII " PARSE ; IMMEDIATE

' " ALIAS C" IMMEDIATE

: (!) [COMPILE] ! ; ( ! für F-PC, W! für Win32For )
: (@) [COMPILE] @ ; ( @ für F-PC, W@ für Win32For )
: (,) [COMPILE] , ; ( , für F-PC, W, für Win32For )

: ANSCOMPARE        ( addr1 len1 addr2 len2 -- f )
```



```
  ROT 2DUP =
  IF DROP COMPARE   ( gleiche Laenge )
  ELSE 2DUP >R >R MIN COMPARE
  ?DUP IF R> R> 2DROP ( unterschiedliche Strings )
       ELSE
         R> R>
         < IF 1 ELSE -1 THEN ( erste Zeichen
                               gleich )
       THEN
  THEN
;

DEFER $=   ' ANSCOMPARE IS $=

CR .( 16-bit Version "F-PC" des String-Packages
      geladen. )
fload string.inc

\ String.F String-Package von Nils Eilers 1998/99,
\ benötigt STRING.INC
\ ************************************************
\ *** ACHTUNG: Unter Win32For ist das Package z.T.
\ noch buggy, da ich nur die 16-bit-Version STRING.
\ SEQ näher getestet habe!  ***
\ ************************************************

\ F-PC Bordmittel:
: $.R   ( addr len n1 -- )
  TUCK UMIN TUCK - -ROT TYPE SPACES ;
: $.L   ( addr len n1 -- )
  TUCK UMIN TUCK - SPACES TYPE ;

: (!) [COMPILE] W! ; ( ! für F-PC, W! für Win32For )
: (@) [COMPILE] W@ ; ( @ für F-PC, W@ für Win32For )
: (,) [COMPILE] W, ; ( , für F-PC, W, für Win32For )

: DARK CLS ;

  DEFER $=   ' COMPARE IS $=

CR .( 32-bit Teil"Win32For" des String-Packages
      geladen. )
fload string.inc
```

# Zeilen und Strings

von

Wil Baden

wilbaden@netcom.com

Nachdruck aus der Forth Dimensions 20 (1998), Heft 2, S.20, mit freundlicher Genehmigung des Autors und der amerikanischen Forth Interest Group.

übersetzt von Fred Behringer

Neue ( NEW ) und alte ( OLD ) Zeilen

Hier zeigen wir eine bequeme Methode, zwei verschiedene Versionen eines Quelltextes zu handhaben. Es kann sich dabei um eine schon bestehende und eine ausgebesserte Version handeln, um ein Standard-Programm und ein leistungsfähigeres, aber umgebungsabhängiges Programm, um Programmteile, die Sie von irgendwoher bekommen haben, und solche, die Sie abgeändert haben, und so weiter.

Statt die alten Zeilen ganz zu entfernen oder auszukommentieren, setzen Sie ihnen ein [O] und einen Zwischenraum voran. Und lassen Sie den neuen Zeilen ein [N] und einen Zwischenraum vorangehen.
Die [N]-Zeilen werden bearbeitet, die [O]-Zeilen werden ignoriert.
Geben Sie 'Reversion ON' ein, bevor Sie die alten und neuen Zeilen bearbeiten, wenn Sie teilweise oder insgesamt zu den alten Zeilen zurückkehren wollen. 'Reversion OFF' macht das wieder ungeschehen.
Um zur neuen Form überzugehen, entfernen Sie mit Hilfe eines Editors alle [N] und alle Zeilen mit [O].

```
VARIABLE Reversion  ( `Reversion ON` für alte Zeilen )

: [N] Reversion @ IF  POSTPONE \  THEN ;
      IMMEDIATE
: [O] Reversion @ 0= IF  POSTPONE \  THEN ;
      IMMEDIATE
```

Sonderzeichen in Strings

Mit S" ccc" können wir eine String-Konstante erzeugen. In diesem String darf aber kein " enthalten sein. Auch keine Steuerzeichen.

Für solche Zwecke schlagen wir in unserer Werkzeugkiste S vor. Wie andere Funktionen in unserer Werkzeugkiste wertet S das erste Zeichen des nächsten Wortes als Begrenzer.

```
      S | Sagen Sie einfach "Nein".| TYPE
```

Um die Steuerzeichen 0 bis 31 wiederzugeben, verwendet S ein ^ vor den 32 bei @ beginnenden Zeichen.  ^? liefert 127 .

Im String-Literal  S " Sagen Sie einfach^M^J^I^INEIN^M^J" werden ein CRLF (Wagenrücklauf und Zeilenvorschub) und zwei tabs (Tabulatorstops) vor dem NEIN eingefügt und wieder ein CRLF dahinter.

Wenn ccc weder " noch ^ enthält, ist  S " ccc"  mit  S" ccc" gleichbedeutend.

```
: ?maximum ( n max -- n ) OVER <
   ABORT" Höchstlänge überschritten " ;
```

'str len caddr >control>' kopiert einen String von str len nach caddr, wobei die Zeichen nach einem ^ in Steuerzeichen umgewandelt werden. Es liefert caddr und die resultierende Länge zurück.

```
: >control>    ( str len caddr --  caddr k )
    0 2>R                                  ( R: caddr k)
      BEGIN  DUP WHILE                      ( str+i len-i)
        OVER C@ [CHAR] ^ CASE? IF
          1 /STRING
          OVER C@  64 XOR
           THEN                             ( . . c)
            2R@ CHARS + C!                  ( str+i len-i)
```

```
            R> 1+  80 ?maximum  >R
            1 /STRING
        REPEAT
    2DROP
     2R>  ;                              ( caddr
k)( R: )
```

Wenn Sie den Namen des von Ihrem System verwendeten Puffers für die String-Literals kennen, setzen Sie diesen anstelle von SBUF ein.

```
80 CHARS BUFFER: Sbuf

: S             ( "<char> ccc<char>" -- caddr k )
     CHAR PARSE ( caddr k) Sbuf >control>
     STATE @ IF  POSTPONE SLITERAL
THEN
 ; IMMEDIATE
```

Anhang

```
: RESERVE ( n -- ) HERE SWAP  DUP
    ALLOT  ERASE ;

: BUFFER: ( n -- ) CREATE  RESERVE ;

: PARAMETER  BL WORD COUNT
    EVALUATE ;

: ??  S" IF " EVALUATE  PARAMETER  S"
    THEN
    " EVALUATE ; IMMEDIATE

: CASE?   ( x y -- true | x false )
    OVER = DUP ?? NIP ;
```

CASE? stammt aus VolksForth. CASE? IF kann durch OVER = IF DROP ersetzt werden. PARAMETER und ?? sind dann überflüssig.

## Forth Interest Group International (FIG USA)

Wollen Sie mit der ganzen Welt verbunden sein und dabei Ihr Englisch perfektionieren? Amerika ist ein wesentlicher Teil der ganzen Welt. Zumindest, was Forth betrifft. Über tausend Mitglieder aus allen Ländern sind bei uns.

Werden auch Sie Mitglied in der Amerikanischen Forth-Gesellschaft (FIG USA).

Für 45 Dollar im Jahr (Studenten zahlen 18 Dollar) bekommen Sie 6 Hefte unserer Vereinszeitschrift Forth Dimensions und genießen auch sonst verschiedene Vorteile. In den Heften erfahren Sie Forth-Neuigkeiten aus aller Welt, neue Produkte, Literatur, Forth-Ideen, fundiertes Wissen, Artikel auch für Einsteiger, Projekte, Leser-Diskussionen, Quelltexte, Hinweise auf Internet-Verbindungen, kostenlose Forth-Systeme und vieles mehr. (Für Übersee-Porto müssen wir leider noch 15 Dollar hinzurechnen).

Unmittelbare Informationen über uns bekommen Sie, wenn Sie auf der Homepage der Deutschen Forth-Gesellschaft "Links zu anderen Forth-Organisationen" und dann "Forth Interest Group (USA)" anklicken.

Ansonsten bekommen Sie Auskünfte über das amerikanische Forth-Büro:

**Forth Interest Group**
**100 Dolores Street, suite 183**
**Carmel, California 93923**
**USA**

oder auch vom Redakteur, Marlin Ouverson, unter der E-Mail-Adresse:

E-Mail: **office@forth.org** oder

# Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

### Forth Dimensions der Forth Interest Group, USA

**Juli/August 1998**

**2 Office News**
**Trace Carter**

Grüße vom FIG-Büro. Abspringende und zukommende Mitglieder halten sich die Waage. 20 Jahre FIG. Auf dieser Seite auch eine Werbeanzeige der Forth, Inc. für SwiftForth für Windows 95/98 und Windows NT. GUI, ANS, DLL, DDE, alles drin.

**4 Editorial**
**Marlin Ouverson**

Hinweis auf den ersten gedruckten begutachteten Artikel (siehe S.3) aus dem Journal of Forth Applications and Research (JFAR). JFAR gibt seine Artikel elektronisch heraus und die Forth Dimensions wird den einen oder anderen davon abdrucken. - Auf derselben Seite auch wieder eine Anzeige der Forth-Gesellschaft zur Mitgliederwerbung. Das Echo ist praktisch gleich Null, aber schön sieht es doch aus - und zum gegenseitigen Aufmerken trägt es ganz gewiß bei.

**5 Finite State Machines in Forth**
**Julian V. Noble**

Ausführliche, 8 Seiten lange Arbeit des Autors des Buches "Scientific Forth" und des Initiators der "Scientific Forth Library" auf dem Taygeta-Server. Alles anhand von Forth erklärt ("Listing 1 bis 18"). Über "Finite State Machines" (FSMs) wurde in der Forth Dimensions schon einiges

gesagt. Die vorliegende Arbeit ergänzt das bisher Gesagte über FSMs (endliche Zustandsautomaten - eigentlich Automaten mit endlich vielen Zuständen - also nicht F(SM), sondern (FS)M). "Forth ist eine überaus gut strukturierte Sprache, in welcher sich FSMs ausgezeichnet und gut lesbar programmieren lassen", sagt der Autor. Wesentlicher Vorteil der FSM-Technik ist die Vermeidung von geschachtelten IF-THEN-Konstrukten. Stattdessen ist von Zustandsmatrizen die Rede. Das Ganze wird erklärt anhand des (Standard?-) Beispiels der Eingabe von Gleitkommazahlen. Weitere Stichwörter: Iterierte FSMs (eine in der anderen), HS/Forth, ANS-Anbindung, nicht-deterministische FSMs, ... warum Zustände durchnumerieren, warum nicht benamen? ... Der äußere Forth-Interpreter ist eine FSM, mit COMPILE und INTERPRET als Zustände.

**13 Writing for Forth Dimensions**
**Marlin Ouverson**

Marlin erhielt eines Tages die "Hinweise für Autoren" einer gewissen Zeitschrift, bei denen man den Eindruck hatte, der Autor müsse es als eine besondere Gnade betrachten, wenn er dort veröffentlichen darf. Nichts da bei Forth Dimensions. Jeder kann mitmachen, jeder ist willkommen, ob Anfänger oder Fortgeschrittener. Dem Anfänger wird redaktionelle Hilfe zuteil. Ein paar grundsätzliche Dinge sollten beachtet werden (sie werden dann kurz erwähnt), sind aber leicht zu erfüllen.

**14 Safer Numeric Input**
**Jerry Avins**

Mit Sicherheitsfragen brauchen sich viele Programmierer sicherlich nicht herumzuschlagen. Aber wenn Ihr Embedded-Control-System eine medizinische Einrichtung oder eine Schwermaschine steuert, werden solche Dinge lebenswichtig. Fachleute auf diesem Gebiet wissen, daß es da viele verschiedene Gesichtspunkte gibt. Im vorliegenden Artikel wird einer davon herausgegriffen, nämlich der, daß das Anzeigenfeld die Daten, die das System verarbeitet, genau wiedergibt.

**18 EXPRESS Factory Control**
**Allen Anway**

EXPRESS ist ein auf Forth aufbauendes Programmier- und Betriebssystem für IBM-Computer, mit dem man große Anlagen oder Fabriken fahren kann. Der Autor berichtet über Erfahrungen beim praktischen Einsatz.

**Mail Order Form**

4 Seiten: Bücher, Disketten, ältere Bände FD, Forth-Systeme, Bibliotheksausleihe. Von Leo Brodies "Starting Forth" kann man noch ein paar Exemplare erwerben.

**20 Lines and Strings**
**Wil Baden**

[O] und [N], um alte und neue Version eines Programmstücks gleich auch für den Probelauf zu unterscheiden. So das eine. Das andere: Mit S" kann man eine Stringkonstante einleiten, die aber den Nachteil hat, " und Kontrollzeichen (0-31) nicht zu verarbeiten. Jeder löst dieses Problem auf seine Weise. Ich (der Rezensent) habe es in meinem Transputer-Forth-System F-TP 1.00 auf meine Weise gelöst. Wil Baden, der anerkannte "Wergzeugkasten"-Konstrukteur, schlägt S " und dann z.B. ^127 vor. Nicht schlecht, schon deshalb nicht, weil man diese Vorgehensweise von vielen anderen Gegebenheiten, DOS-Batch usw., her in Erinnerung hat und sie sich also leicht merken kann.

**21 Local Variables for Misers**
**Neil Bawd**

Über lokale Variable ist schon viel gesagt worden. Diesmal für "Geizige". Ein ganz kleiner Vorschlag.

**22 Character Tests**
**Wil Baden**

Wieder ein paar Worte des bekannten Autors zur sinnvollen Erweiterung von ANS-Forth. SKIP , SCAN und -TRAILING wurden 1982 von Klaus Schleisiek (!) eingeführt und von Laxen and Perry für F83 übernommen. Der Autor möchte Strings auch über Leerzeichen und Steuerzeichen hinweg durchkämmen können. Er schlägt etwa 14 Worte vor (ein paar sind im Text verklausuliert enthalten) und formuliert weitere 10 aus seinem Artikel im letzten Heft mit deren Hilfe neu.

**25 Forth to the IRS**
**Len Zettel**

Einkommensteuererklärung scheint in Amerika mindestens so kompliziert zu sein wie bei uns. Der Autor verbindet den Horror, den er Jahr für Jahr davor hat, mit seiner Liebe zu Forth. Die fiskalischen Bestimmungen sind in ihrem Bestreben, genau zu sein, so unverständlich umständlich geschrieben, daß man nur noch Wort für Wort nach Forth zu übertragen braucht, um ein ebenso genaues und hoffentlich nicht ganz so unverständliches Programm zu erhalten.

**26 EuroForth 1998 Conference**
**Paul E. Bennett**

Was für eine Arbeitserleichterung für den Rezensenten! Ich brauche über diesen Artikel nichts mehr zu sagen. Es ist genau derselbe Artikel wie der aus der Forthwrite 99. Halt, einen Unterschied habe ich (präzise, wie unsereiner nun mal ist) festgestellt: Im englischen Artikel wird "travelling" mit zwei Ell geschrieben, im amerikanischen wurde dem Autor ein Ell gestrichen. Man verzeihe mir diese Bemerkung, die

losgelöst vom vorliegenden Artikel durchaus einen ernsthaften Hintergrund hat, aber: Ein bißchen Spaß muß sein. Was nützt uns unsere ganze Debatte um unsere Schreibreform? Das hier sind unsere Schwierigkeiten. Für wen sollen wir uns denn im globalen Dorf entscheiden?

**30 Finanzielle Anerkennung für Zeitschriftenbeiträge über Forth**

Jeder Autor, der eine forth-bezogene Arbeit von mindestens einer Seite Länge in einer Zeitschrift oder in den Tagungsbänden einer Nicht-Forth-Konferenz veröffentlicht hat, bekommt eine Gratis-Mitgliedschaft bei der Forth Interest Group (USA) für ein (ganzes) Jahr. Artikeln in einer anderen Sprache als Englisch muß bei der Meldung an die Forth Interest Group eine Übersetzung beigefügt werden. "Briefe an die Redaktion" (an eine andere als die der Forth Dimensions) eines "Periodicals" werden mit 10 Dollar honoriert, die sich auf bis zu einem Mitgliedsbeitrag (45 Dollar) aufsummieren dürfen. Der Rezensent: Es steht nirgends, daß das alles nicht auch für unsereins gilt. Spitzfindige Frage: Ist die VD eigentlich ein "Periodical"?

**31 Freeware & Shareware**

JForth hat nun auch einen Zugang über home.tampabay.rr.com/jforth/

**32 Two Problems in ANS Forth**
**Thomas Worthington**

Der Autor steht dem ANS-Forth (im Vergleich zu F83) kritisch gegenüber. Schwierigkeiten mit colon-sys auf dem Steuerfluß-Stack (meist der Datenstack) und die Unmöglichkeit, den Eingangsstrom in einen willkürlich wählbaren Speicherbereich zu lenken. Der Autor unterbreitet Lösungsvorschläge - in ANS-Forth.

**FORML Conference**
**Marlin Ouverson**

Aufruf zur 20. FORML-Tagung, diesmal in Pacific Grove, California, 20.-22. November 1998, inzwischen schon Vergangenheit. 14 Hauptvorträge. Ich greife willkürlich einen heraus: "Open Network Forth - Control system for the Munich accelerator facility" von Ludwig Rohrer und Heinz Schnitter.

## September/Oktober 1998

**2 Office News**
**Trace Carter**

Alles wird teurer, Bücher, Disketten, ältere FD-Ausgaben auch. Wer sie noch zum alten Preis haben will, sollte sich sofort entscheiden, bevor die nächste FD mit den neuen Preisen erscheint. - FIG-Mitglieder können eine E-Mail-Adresse der Form mein-name@forth-org beantragen. Näheres im Forth-Büro.

**4 Editorial**

Marlin Ouverson macht auf die "Design Your Own Processor"-Tools aufmerksam: http://www.dnai.com/~jfox/fpgakit.htm . Außerdem liest man auf dieser Seite wieder unsere Mitgliederwerbung für die Forth-Gesellschaft. Ich finde das sehr erwähnenswert, nicht so sehr deshalb, weil wir uns dadurch mehr amerikanische Forth-Freunde als Mitglieder der FG versprechen, sondern wegen der Aufmerksamkeit, die wir dadurch auf uns ziehen. Wir werden dementsprechend die Anzeige der amerikanischen Freunde weiterhin regelmäßig in der VD bringen.

**5 Porting hForth to the StrongARM SA-110RISC Processor**
**Neal Crook**

Ein sehr interessanter Artikel von 6 Seiten. Diejenigen, die immer schnell mit dem Urteil "das Rad neu erfunden" bei der Hand sind, werden anderer Meinung sein. Ich empfinde mit dem Autor mit. Die ganze Geschichte seines Vorgehens. Selbstverständlich kommt einem vieles bekannt vor: ob eForth auf einen Simulator und dann auf einen ARM610 transportieren oder dann doch zu ANS-Forth über den Weg von hForth - ist ja alles egal, sagen die einen (die meinen, schon alles zu wissen). Mir macht es Spaß zu sehen, wie jemand das Rad auf seine Weise erfindet. Jeder Prozessor ist anders - vieles bleibt sich jedoch überall gleich. Generelle Regeln für den Neuaufbau oder die Anpassung eines Forth-Systems scheint es nicht zu geben. In gewisser Weise natürlich schon - sonst hätten ja solche Artikel keinen Sinn. Man muß sie sich aber selbst herauslesen, so wie ein Kind seine Sprache lernt, und sie seinen eigenen schon gefestigten Vorstellungen anpassen. Die Prozessoren im Artikel sind mir fremd, die Vorgehensweise des Autors ist mir aber irgendwie vertraut - und ich lerne, vergleiche und lerne. "Forth", so der Autor, "kann als ein spezieller Fall eines Programms betrachtet werden, das sich selbst modifiziert: Ein Programmteil, der gerade ausgeführt wird, nimmt Veränderungen an seiner eigenen Befehlsfolge vor." Der Autor, das merkt man, hat sich sehr mit der Materie beschäftigt. Hier noch die Quellen, von denen man sich die im Artikel besprochenen Systeme herunterladen kann: http://www.taygeta.com/forthcomp.html oder ftp://ftp.taygeta.com/pub/Forth/Compilers/native/dos/hForth .

**11 The Stuttering Context Switch**
**Martin Schaaf**

Kontext-Umschaltung in Forth. Engpaß Datenschaufeln vom und zum Stack. Der Autor schlägt zwei parallel arbeitende Prozessoren vor. Der eine ständig arbeiten (schaufeln), der andere immer fleißig - sie kheern zamm. Der Autor bittet um Nachsicht: Er wollte den Pentium nicht neu erfinden. Seine Ideen stammen aus einer Zeit (vor 15 Jahren) als Multitasking noch nicht in Sicht war.

**12 Linearizing a Thermocouple with Two-Step Interpolation**

**Jerry Avins**

Temperaturregelung in einem kleinen industriellen Ofen. In nur ganz geringem Maße nichtlinear. Funktionsapproximation. Normalerweise Auswertung eines Polynoms verlangt. In solchen Fällen üblich Polynom neunten Grades! Viel zu kompliziert. Ausreichend stückweise Approximation durch Polynom zweiten Grades (Parabel) in Festkomma-Arithmetik. Alles in Forth. Arbeitet auch für kompliziertere Funktionen. Beispielsweise Sinus und Cosinus, wenn nur Rechenschiebergenauigkeit verlangt wird. - Daß der Autor in Fahrenheit denkt, wo wir in (ganz?) Europa Celsius verwenden, zeigt, daß wir von einem wirklich globalen Dorf noch meilenweit (Verzeihung, kilometerweit) entfernt sind.

**15 Stacks - A Crossword**
**Neal Bridges**

Aus dem Internet: Ein Kreuzworträtsel: Die Stackbelegung in der Art von ( x1 x2 -- x1 x2 x1 ) wird für 11 Forth-Worte angegeben, die Worte selbst (senkrecht und waagerecht) werden gesucht. Hier meine Lösung: NIP SWAP 2SWAP 2DUP DROP DUP 2DROP ROT TUCK 2OVER OVER . Ganz leicht - für den, der's kann. Interessant, wie viele Wege es gibt, die Aufmerksamkeit des Anfängers auf sich zu lenken!

**16 Growth and Changes**
**Skip Carter**

Ein Aufruf des Präsidenten von FIG. Artikel werden gebraucht und Buchautoren werden gesucht. Anfang 1998 waren es 900 Mitglieder, jetzt sind es etwa 700. Im Juni 1999 wird ein neuer Vorstand gewählt (7 Mitglieder im Directorial Board). Die gesamte FIG wird von Freiwilligen gelenkt. Es gibt keine bezahlten Posten. Es wird eine neue Art von Mitgliedern eingeführt, E-Mitglieder: solche, die die Forth-Dimensions-Hefte (nur) elektronisch (in PDF-Format) zugestellt bekommen. Mehr darüber im nächsten Heft.

**17 ANS Appendix to Finite State Machines in Forth**
**Julian V. Noble**

Das Programm zum gleichnamigen Thema desselben Autors über endliche Zustandsautomaten aus der letzten FD, diesmal nach ANS-Forth übertragen. Außerdem ein paar Korrekturen.

**19 A Forth Switchblade**
**Rick VanNorman**

"Immer auf der Suche, nie zufrieden, trage ich immer neue Forth-Werkzeuge und -Verfahren zusammen." Sehr sympathisch, was Rick, der Schöpfer eines zu Unrecht (der Rezensent) wenig beachteten 32-Bit-Forths, das unter anderem auch in der DOS-Box von Windows 3.11 läuft, da schreibt. Es geht um "bedingte Verteiler" (Worterfindung des Rezensenten), "Switcher". (Schalter" ist nicht der richtige Ausdruck. Eher so'ne Art Hebdrehwähler, Umschalter, Auswähler - der Rezensent). Rick gibt Beispiele aus C an. In OC-CAM macht sich "CASE" auch gut (der Rezensent). "Eakers Case", so Rick, "kann bis zum Erbrechen optimiert werden" (und sieht wirklich schön aus - der Rezensent), "warum also weiterhin Gedanken an das Auswählen einer bestimmten Reaktion auf das Vorliegen einer gegebenen Bedingung verschwenden?". "CASE kann", so Rick, "wenn es einmal definiert ist, nicht mehr leicht verändert und erweitert werden." Rick arbeitet (in diesem Bericht) an einem Forth für Windows unter SwiftForth und möchte sein "CASE-Konstrukt" jederzeit, auch mitten im Compilieren oder beim Austesten von der Tastatur aus, verändern und anpassen können. Sehr interessant! Der Rezensent: Man kann über "CASE" die Nase rümpfen. Es ist noch lange nicht alles gesagt. Schon mal CASE für Stringvergleiche entwickelt? Ein SOF ("String-OF") zusätzlich und als Konkurrenz zu OF macht es möglich und läßt sogar Mischungen zu! Fragen Sie mich.

**23 Point and Do**
**Richard W. Fergus**

Der Bildschirm wird in eine Anzahl von Blöcken aufgeteilt. Innerhalb eines jeden solchen Blocks können mit Hilfe der Maus eine Reihe von Funktionen an den im Block enthaltenen Bildelementen ausgeführt werden. Der Autor hat das Programm für sein "Embellished Pygmy Forth" geschrieben.

**26 Number Conversion and Literals**
**Wil Baden**

NUMBERS? und NUMBER aus Brodies "Starting Forth" werden mit Erweiterungen und Verbesserungen nach ANS-Forth übertragen.

**29 Only Standard Definitions**
**Wil Baden**

Eine Wortliste und ein kleines Programm, mit dessen Hilfe man sein System daraufhin automatisch überprüfen kann, ob es nur Standard-Worte enthält.

**34 URLs - a selection of Web-based Forth resources**

Vierundzwanzig HTTP:-Adressen mit Forth-Bezug. Wir sind nicht dabei. Aber es ist ja auch nur eine "Auswahl".

## VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

**Nr. 12, Dezember 1998**

Das Vijgeblaadje hat diesmal doppelten Umfang (8 A4-Seiten statt der sonst üblichen 4 Seiten). Die eine Hälfte davon ist dem Egel-Werkboek-Projekt gewidmet. Und auch in der anderen, der "normalen", Hälfte nimmt das Egel-Werkboek den meisten Platz ein.

**Egel werkboek**
**Leendert van den Heuvel**

Das Projekt wurde vor einem Jahr von Willem Ouwerkerk aus der Taufe gehoben. Gedacht ist es für Forth-Freunde, die auch mit dem Lötkolben umgehen können, für wirklich echte Forth-Freunde also. Angewachsen ist es von urspünglich 7 auf inzwischen 21 Einzelprojekte, die sich alle um den Mikrocontroller AT89Cx051 von Atmel aus der Familie 8051 ranken. Das Ganze ist das Werk von sechs Enthusiasten, die sich allmonatlich im Bahnhofsrestaurant von Utrecht bis über die Sperrstunde hinaus trafen. Wir werden wahrscheinlich noch mehr über dieses Projekt hören. Für diejenigen, die ihre Neugier schon jetzt nicht zügeln könne: Willem Ouwerkerk <voorz@forth-gg.hobby.nl> gibt sicher gern Auskunft.

**FiFo**
**Albert Nijhof**

LiFo (last in - first out) ist von den Stacks her geläufig. Der Autor bietet eine Implementation einer Warteschlange, FiFo (first in - first out), an (3 Colon-Definitionen). Außerdem noch ein "Sammelbecken" (unregelmäßiger Zugang - paketweiser Weggang) in weiteren 5 Colon-Definitionen.

**Nr. 13, Februar 1999**

Es wird ein Artikel von Wil Baden aus der Forth Dimensions über das Multiplizieren langer Zahlen (ins Holländische) übersetzt. Den Artikel habe ich bereits andernorts im Original besprochen. Sodann werden noch ein paar Bemerkungen über das Igel-Projekt gebracht. Das Igel-Projekt habe ich bereits und werde ich noch einmal ausführlich besprechen.

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 20 Gulden 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:

**Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>**

Oder überweisen Sie einfach 20 Gulden auf das Konto 5253572 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden, Willem Ouwerkerk, zu wenden.

# Das Igel-Arbeitsbuch

**aus dem Holländischen übersetzt von Fred Behringer und mit freundlicher Genehmigung der HCC-Forthgebruikersgroep (Vorsitzender und Initiator des Projekts: Willem Ouwerkerk) wiedergegeben**

Von der Forth-gebruikersgroep auf dem Treffen 1998 des Hobby-Computer-Clubs (HCC) vorgestellt: Hard- und Software-Projekte für die Mikroprozessor-Familie 8051 von Atmel: AT89Cx51.

***Stichworte: ByteForth, Hardware, Software, Mikrocontroller, Atmel AT89Cx51, 8051***

**Das Projekt.** Ein "Igelchen" ist eine Karte mit einem Mikrocontroller und anderen kleinen Stacheln drauf. Um es zum Leben zu erwecken, müssen Sie:

1. es mit der Außenwelt verbinden (Ein- und Ausgabe),
2. ihm das logische Denken beibringen (programmieren) und
3. es füttern (Speisung über Batterie).

Mit Sensoren für die Eingabe fängt der Igel Signale auf. Die werden vom Prozessor des Mikrocontrollers verarbeitet. Das Programm bestimmt, was für Aktionen daraufhin an der Ausgabe stattfinden.

Zur Umkehrung der Drehrichtung eines kleinen Motors reicht ein Schalter und, falls nötig, ein Relais aus. Da brauchen Sie keine Elektronik. Aber zur Regelung der Geschwindigkeit oder zum Antreiben eines Schrittmotors wird schon etwas mehr benötigt.

Dank der Intelligenz der Prozessoren können mit wenigen Extrabauteilen allerlei Anwendungen erstellt werden. So zum Beispiel ein Sender und ein Empfänger für eine Infrarot-Fernbedienung.

Sie können sich selbst einen Igel bauen, für wenig Geld. Und das bieten wir an.

Das Igel-Arbeitsbuch ist als Lehr- und Ideenbuch zum Bauen eigener Hard- und Software-Anwendungen gedacht und enthält

- Schaltbilder, Platinen-Layouts und Stücklisten für die Hardware

- Listings für die Software (auch auf Diskette)
- Ergänzungen mit weiterführenden Informationen .

Das Arbeitsbuch enthält eine große Anzahl von einfachsten Anwendungen (für die Ports, RS-232, I2C, ADC usw.), die Sie als Bausteine für Ihre eigenen Neuentwürfe verwenden können.

Die im Arbeitsbuch behandelten Projekte wurden mit handelsüblichen Bauteilen aufgebaut.

Wollen Sie darauf mit weiteren Eigenentwicklungen aufbauen, dann finden Sie alles, was dazu nötig ist, in diesem Arbeitsbuch und im ByteForth-Handbuch. Die verwendete Programmiersprache ist Forth. Ein (Schnell-)Kurs in Forth wurde in das Handbuch mit aufgenommen.

Wir haben folgende Varianten zusammengestellt:
1. Das komplette Anfängerpaket
   a. mit der AF-Karte
   b. mit der F+-Karte
2. wenn Sie schon eine F+-Karte haben
3. wenn Sie schon eine B+-Karte haben
4. die PC-Version (mit Emulator) .

In den folgende Angaben bedeutet AFBP beziehentlich die AF- , F+- , B+- , PC(Em.)-Version. Preise für Fgg-Mitglieder gefolgt von den Preisen für Nicht-Mitglieder (f = holländische Gulden).

AFBP Das Igel-Arbeitsbuch ... f 25,- f 30,-
- einschließlich Listing auf Diskette

P Das ByteForth-System der Version 1.70pc ... f 140,- f 155,-
- das ByteForth-Handbuch 1.70pc
- Server.exe und ByteForth 1.70pc
- Flash-Programmierer RS-FLAP
- ein programmierter AT89C2051 für den RS-FLAP
- zwei Mikrocontroller AT89C2051
- eine AT-51-Experimentierkarte

AFB Das ByteForth-System der Version 1.70 ... f 100,- f 110,-
- das ByteForth-Handbuch 1.70
- Server.exe und ByteForth 1.70
- Flash-Programmierer FLAP
- zwei Mikrocontroller AT89C2051
- eine AT-51-Experimentierkarte

Zur Auswahl: AF oder F+

A AF-Karte, fertig bestückt, einschließlich 8052-EPROM ... f 200,- f 225,-
und mit dem 8052-Handbuch (siehe nächste Zeile!)

A AF-Karte mit ByteForth 1.70 wie obenstehend ... f 255,- f 280,-
(also zwei in einem), jedoch ohne 8052-Handbuch

F F+-Karte, Platine (Paul Wiegmans)

BB+-Karte, nicht lieferbar (siehe AF-Karte)

FB 8052-ANS-Forth in EPROM ... f 90,- f 100,-

AFBP Extra-Mikrocontroller AT89C2051 ... f 15,- f 17,50

AFBP Extra-Experimentierkarte AT-51 ... f 15,- f 17,50

AFB FAT-Stromversorgungsplatine (nicht lose lieferbar) ... f 8,- f 10,-

A AF-Leuchtdiodenkarte, fertig bestückt ... f 20,- f 25,-

AFBP Platine für das Projekt 7-Sgm-LEDs ... f 20,- f 25,-
- einschließlich Dokumentation

AFBP Platine für das Projekt I2C ... f 25,- f 30,-
- einschließlich Dokumentation

Versandkosten
- bei Vorauszahlung ... f 15,-
- per Nachnahme ... f 25,-

**Was brauche ich noch?** Einen Heimcomputer mit einem RS232-Anschluß. Der Server wurde für den PC geschrieben, aber jedes andere System mit einem eigenen Kommunikationsprogramm ist, mit geringfügigen Einschränkungen, geeignet. Die (neueste) PC-Version mit Emulator funktioniert auf den "Kompatiblen".

**Wie komme ich an mehr Information?** Oder an wen kann ich mich wenden, wenn ich etwas (könnte ja sein) nicht kapiere?

1. Kommen Sie zu unseren Treffen, siehe Forth-gg
2. Fido-BBS, Tel.: 026 4422 164
3. E-Mail für das Igel-Projekt: voorz@forth-gg.hobby.nl
   F+ (Paul Wiegmans): spectral@simplex.nl
   Alles über die Fgg: voorz@forth-gg.hobby.nl
4. Telefonieren Sie, schreiben Sie uns.
5. Lesen Sie auch Computer!Totaal unter Benutzergruppen.

**Die Programmiersprache Forth.** Forth ist eine High-Level-Programmiersprache. Geben Sie auf Interpreter-Niveau eine Routine (Colon-Definition) ein, dann wird diese direkt compiliert. Das heißt eigentlich, daß Sie die Sprache selbst um ein neues Wort erweitern. Die kompilierten Routinen können sofort nach Eingabe verwendet werden. Insbesondere kann man sie sofort Wort für Wort austesten, ohne sich um den Rest des Programms kümmern zu müssen. Das ist das, was man einen interaktiven Compiler nennt.

Auf die Stacks kann unmittelbar zugegriffen werden und Assembler-Code läßt sich direkt in das Listing einauen. Mit Forth haben Sie auch unmittelbaren Zugriff auf die Hardware. Das ist der Grund, weshalb man Forth sehr häufig bei Steuerungsaufgaben und Mikrocontrollern einsetzt.

Im Arbeitsbuch verwenden wir ByteForth, ein 8-Bit-Forth, das auch 16-Bit-Werte verarbeiten kann und weitestgehend mit dem ANS-Forth-Standard übereinstimmt. Für unsere Anwendungen haben wir ein paar zusätzliche Werte eingeführt,

so daß der Mikrocontroller mit leicht merkbaren Namen angesprochen werden kann.

**Das ByteForth-Handbuch.** Das ist ein Buch, das die Verwendung der Software (Server usw.) beschreibt und das als Lehr- und Handbuch für das Schreiben von Programmen dienen soll. Es enthält einen Schnellkurs in ByteForth.

**Das Igel-Arbeitsbuch.** Im Igel-Arbeitsbuch werden Elektronik-Bauprojekte beschrieben, die als Ausgangspunkt für Eigenentwicklungen, wie Steuersysteme, Meßsysteme usw., genommen werden können. Es enthält 21 Projekte, die ein weites Gebiet überstreichen.

Das ist das erste Arbeitsbuch. Haben wir noch ein weiteres zu erwarten? Ja, es wird eine neue Arbeitsgruppe für ein zweites Arbeitsbuch eingerichtet, in dem neue, weiterentwickelte Projekte vorgestellt werden. Beispiele: Roboterarm, PIR/Sensor, Steuerung einer Satellitenschüssel. Wenn Sie Ideen für interessante Projekte haben, liegen Sie bei uns richtig: Auf den HCC-Treffen. Oder über E-Mail, oder BBS oder ...

**AT89C2051.** Dieser Mikrocontroller von Atmel gehört zur Familie 8051. Der Prozessor enthält 2KB Flash-Speicher, 128 Byte RAM, Timer, E/A-Ports und anderes. Das alles sitzt in einem einzigen 20-Pin-Chip-Gehäuse. Im Arbeitsbuch wird der AT89C2051 in 12MHz-Ausführung verwendet. Der eingesetzte 6MHz-Kristall ist allerdings für alle Anwendungen mehr als schnell genug: 500000 Befehle pro Sekunde. Man kann aber auch auf 12 MHz (und sogar auf 24 MHz) gehen. Ohne Schwierigkeiten lassen sich auch der AT89C1051 und der neue AT89C4051 einsetzen. Und grundsätzlich auch viele andere Controller aus der Familie 8051.

**Flash-Speicher.** Ein Flash-Speicher ist ein programmierbarer Speicher (PROM), der in seiner Gesamtheit elektrisch gelöscht (EPROM) und wieder beschrieben (Flash-EPROM) werden kann. Sein Inhalt bleibt danach erhalten, auch wenn die Speisespannung ausgeschaltet wird. Der 2051 hat 2K an Flash-Speicher an Bord. Dieser Flash-Speicher kann mindestens 1000 mal gelöscht und wieder beschrieben werden.

**Flash-Programmierer.** Ein Flash-Programmierer kann einen Chip mit Flash-Speicher wie den AT89C2051 programmieren. FLAP ist der Programmierer für den Einsatz bei AF/F+/B+-Karten. RS-FLAP ist der Programmierer für den Einsatz bei den PC-Versionen.

**Die Forth-gg.** Die Forth-gg ist eine Anwender-Gruppe innerhalb der HCC. Bei ihren Aktivitäten wendet sie die Programmiersprache Forth an.
Ziele
- Den Einsatz von Forth propagieren
- Informationen über Forth liefern
- Forth (die Sprachversionen) und Anwendungen verbreiten
- Die Mitglieder bei der Anwendung von Forth unterstützen
  Aktivitäten
- Treffen (am zweiten Samstag eines jeden geraden Monats)
- Kurse und Arbeitsgruppen
- Forth FIDO BBS 026 4422 164, Tag und Nacht erreichbar
- Bibliothek (Informationen auf dem BBS)
- Herausgabe des Vijgeblaadjes.

**Eine Übersicht über das Angebot der Fgg.** Die Preisangaben beziehen sich auf Mitglieder, bzw. Nichtmitglieder.

**Software.** CHForth ... f 60,- f 67,50 , ein Mehrsegment-ANS-Forth für den PC/AT, mit englischsprachigem Handbuch. JINForth ... f 35,- f 40,- , ein 32-Bit-ANS-Forth für den Atari-ST auf 3,5"-Diskette. C64-pre-ANS-Forth ... f 15,- f 20,- , ein 16-Bit-beinahe-Ans-Forth auf Diskette.

**Hardware.** Bamboe ... f 50,- f 55,- , eine 16-Bit-E/A-Karte, über die Parallel-Schnittstelle ansteuerbar, für verschiedene Computer geeignet. Bamboe und Kurs ... f 60,- f 67,50 , Bamboe, eine in Kursform gegossene Beschreibung der Bamboe-Hardware, die Forth-Software und eine Anwendung.

**Firmware.** 8052-ANS-Forth-EPROM ... f 90,- f 100,- , ein EPROM mit einem 16-Bit-ANS-Forth für die AF-, F+-, oder B+-Karten mit einem 80C535 (bei der AF-Karte eingeschlossen), mit einem umfangreichen Handbuch (englisch) und Beispielen. Auch für den 80C32 und den 80C552 lieferbar. Das ist der Nachfolger vom 8051-ANS-Forth.

**Dokumentation.** ANS-Forth-Software-Kurs ... f 35,- f 40,- ; Kursunterlagen für den ANSI-Forth-Standard. Vorkenntnisse über Forth werden nicht benötigt. Holländisch. ANS-Forth-Hardware-Kurs ... f 15,- f 17,50 .

**Diverses (nur für Mitglieder).** Forth-ANSI-Standard (X3.215-1994) auf Diskette ... f 7,50 . Frühere Vijgeblad-Hefte (beschränkter Vorrat, per Stapel) ... f 15,- . Vijgeblaadje-Broschüren (soweit vorrätig) ... gratis.

**Bestellungen.** Willem Ouwerkerk, E-Mail: voorz@forth-gg.hobby.nl , Boulevard Heuvelink 126, 6828 KW Arnhem, Tel.: 026 443 1305 . Preise zuzüglich Versandkosten, einschließlich Mehrwertsteuer. Bei Vorauszahlung (Girokonto 5253572 zugunsten HCC Forth-gg zu Amsterdam) ... f 15,- . Bei Nachnahme ... f 25,- .

**Mehr über den Igel.** Im Arbeitsbuch wurden 21 Projekte ausgearbeitet. Die ersten 7 bilden die Einleitung. Sie werden alle sieben zusammen auf einer einzigen Experimentierkarte aufgebaut. Die restlichen Projekte sind noch nützlicher. Beispiele: Relais, Gleichstrommotor, Servo, Schrittmotoren, RS-232, LCD, Infrarot-Fernbedienung, I2C.

Die meisten Projekte können auf einer Experimentierkarte (AT-51) aufgebaut werden. Für zwei Projekte (7-Sgm-Display und I2C) wurden Extrakarten entwickelt und diese sind auch lieferbar.

Das Arbeitsbuch wird mit den hierfür speziell zusammengestellten Experimentier-, Test- und Entwicklungspaketen benötigt (AF-, F+-, B+- oder PC-Version). Diese Systeme werden mit Dokumentation geliefert. (Ob die Anwendungen ausgetestet sind? Natürlich!)

**AT-51.** Die AT-51 ist eine Experimentierkarte, die zum Einsatz bei den Projekten aus dem Arbeitsbuch entworfen wurde.

**AF.** Die AF-Karte ist ein Neuentwurf für den 8051: gedacht für den SIEMENS-Prozessor 80C535(a). Der Entwurf stammt von Bas Boetekees. Sie stellt eine neue, verbesserte Version der B+-Karte dar. Die AF-Karte wird zum Programmieren des Mikrocontrollers AT89C2051 auf der Flash-Programmer-Karte FLAP benötigt. Sie wird fertig aufgebaut geliefert. Die Programme sollten sich auch auf der AF-Karte, fast in Echtzeit, austesten lassen, bevor sie für den AT89C2051 umgeschrieben werden. Wir liefern keine Anpassung, da das Direktprogrammieren des gewählten Mikrocontrollers so einfach ist. Und weil der Chip das sicher 1000 mal vertragen kann. Die AF-Karte ist damit also auch ein eigenständig arbeitender Mikrocontroller mit eigenen Extramöglichkeiten. Die hierzu benötigte Dokumentation ist das (neue) 8052-ANS-Forth-Handbuch, das auch dem AF-Paket beiliegt.

**F+.** Auch die F+-Karte hat als 8051-Prozessor den 80C535 von SIEMENS. Es handelt sich dabei um eine Neuentwicklung von Paul Wiegmans. Sie enthält auch I2C mit einem Echtzeit-Taktgeber. Im Igel-Projekt verhält sie sich genauso wie die AF-Karte. Sie wird als unbestückte Platine geliefert.

# Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

## FORTHWRITE der FIG UK, Großbritannien

### Nr. 100, Januar 1999 (erweiterte Jubiläumsausgabe)

54 Seiten, also wesentlich umfangreicher als gewöhnlich. Im Editorial werden die internationale Zusammenarbeit, die Anfänge der FIG UK und die drei laufenden Projekte (auch mit internationaler Beteiligung) als Schwerpunkte hervorgehoben.

**2 Forth News**

Drei Seiten Neuigkeiten aus dem Internet.

**5 Writing the World (2)**
**Paul E. Bennett**

Zweiter Teil der von Paul begonnenen Serie: Zeitbedingte Ausgaben, Zeitgeber (Timer) programmieren (in Forth), Analogausgaben (mit Schaltbildern).

**9 Dutch Forth Users Group**

Eine Mitgliederwerbe-Anzeige unserer holländischen Forth-Freunde. Im nächsten Heft sind wir wieder dran.

**10 Working with Wordlists**
**Jack Brien**

Ein sehr interessanter Artikel über Wortlisten und Vokabulare: Der ONLY-ALSO-Mechanismus von Forth-83 und die Frage "Wie wär's mit einem besseren Weg?"

**15 Editorial**
**Chris Jakeman**

Noch ein Editorial: Bill Powell schrieb 1979 einen Brief an die Wireless World. Damit begann Forth in England: Im November darauf wurde die FIG UK gegründet. 20 Jahre ist das her. Am 13. November ab 17.00 im Hotel Portobello Gold, London, nicht allzu weit von Paddington Station (von Miss Marple her bekannt), wird gefeiert.

**16 The Small Platoos**
**Chris Hainsworth**

Der Vorsitzende der FIG UK berichtet vom allerersten Forth-Heft, vor 20 Jahren. Der allerallererste Artikel stammte von Bill Powell. Bill erfand TO . (Wie viel habe ich, der Rezensent, schon über TO gelesen. Bill Powell also. Interessant. - Es geht dabei um Stack-Forthens Verhältnis zu "Variablen".) "Wieso konnte Forth und Forthwrite 20 Jahre lang überleben?", fragt sich Chris Hainsworth. "Forth ist ganz leicht und zugleich ganz schwer", gibt er sich selbst zur Antwort.

**20 The Birth of FIG UK**
**Bill Powell**

Hier wird ebendieser Artikel von Bill wiedergegeben. Bill verziert ihn mit ein paar erklärenden Sätzen: Seinerzeit Entwicklung von Verstärkern für Telephonverbindungen unter dem Atlantik hindurch von England nach USA. Vier Monate mit Forth beschäftigt. Reservierte Haltung bei seinen Kollegen. PDP11-Minicomputer mit 64 K RAM. ... Später eine ganze Reihe weiterer Unternehmungen mit Forth. ... empfehlenswert, dieses Auf und Ab eines der ersten Forth-Enthusiasten.

**24 ANS File Words for Turbo Forth: - a first step**
**Fred Behringer**

Es geht um "(", das in ANS-Forth bei Dateieingaben bis zum Ende einer per INCLUDE <name.ext> hinzugeladenen Datei wirken muß, was es aber in Turbo-Forth nicht tut.

**26 Vierte Dimension**

Der Redakteur dankt Alan Wenham für seine Bemühungen, den Inhalt unserer VD-Hefte den englischen Lesern nahezubringen. "Aber es kam bisher noch keine Reaktion" von Seiten der FIG-UK-Leser, weder Zustimmung noch Ablehnung. "Der Aufwand ist also nicht länger gerechtfertigt", sagt Chris, der ungeheure Aufwand nämlich, der darin besteht, je-

weils immer ein ganzes Heft in einer fremden Sprache, in diesem Fall die unsere, aufmerksam durchzulesen und zu besprechen. "Die FIG UK wird", so Chris, "diese Dienste (an ihre Leser) ab jetzt stark einschränken."

Das macht mich (den Rezensenten) natürlich auch nachdenklich. Warum rezensiere ich selbst eigentlich alle drei nichtdeutschen Forth-Zeitschriften laufend? Reaktionen kommen bei uns (selbstverständlich?) auch keine, weder Zustimmung noch Ablehnung. In der Wahltheorie (Unmöglichkeitssatz von Arrow und Ähnliches - ich habe jahrelang darüber Vorlesungen gehalten) wird vom (indirekt) Befragten als Inbegriff des "rationalen" (d.h. des vernünftigen) Handelns das Vorliegen einer konnexen Quasiordnung als Präferenz/Indifferenz-Relation erwartet. Diese läßt drei Möglichkeiten der Entscheidung zu: (1) Zustimmung, (2) Ablehnung, (3) beide Möglichkeiten (alternative "Optionen") als gleichwertig betrachten. Die vierte, von der Logik her denkbare, Entscheidungsmöglichkeit, die des "ich kann oder/und will mich nicht entscheiden, laßt mich mit eurer Fragerei in Ruhe" (Nichtwähler-Verhalten) ist in einer konnexen Quasiordnung (Inbegriff der Rationalität) eben wegen der Konnexität der Relation nicht möglich und wird in wahl- und entscheidungstheoretischen Fragen meist gar nicht erst in Erwägung gezogen (Rationalitätsprinzip - zweite "demokratische" Grundregel von Arrow). Soweit die Theorie. In der Praxis unterscheiden wir uns kaum von unseren englischen Forth-Freunden - und machen regen Gebrauch von der "vierten" Möglichkeit, der Stimmenthaltung.



**27 Vierte Dimension '98 No 4**
**Alan Wenham**

Gut hat der Alan das wieder gemacht, wirklich gut, unseren englischen Forth-Freunden die Aktivitäten aus der VD (die ja bei weitem nicht alles aus der FG sind) nahezubringen. Schade, daß sich da jetzt eine Zäsur abzeichnet. Wirklich schade.

**29 Editorial**
**Chris Jakeman**

Und noch'n Editorial. Es wird ausführlich über die drei laufenden Projekte der FIG UK berichtet: "Erste Schritte in Forth" von David (Dave) Pochin. Abgedruckt sind zwei von Daves Webseiten: http://www.sunterr.demon.co.uk/guide.htm und ..... /header.htm . Es scheint sich hauptsächlich um eine web-basierte Einführung in Win32Forth zu handeln. (Wir sind also nicht die einzigen, die über die Aktivitäten von Marc Petremann nicht Bescheid wissen. - Wenn sich jemand für Marcs Einleitung in Win32Forth interessiert, werde ich darüber berichten.) Zweites Projekt ist ein Hardware-Projekt unter der Leitung von Jeremy Fowell. Chris druckt auf Seite 32 einen Beitrag von Jeremy ab: "FIG UK Hardware Project". Ein Baukastensystem von Einzelteilen, die es gestatten, einen 8-Bit-Mikrocontroller-Einplatinen-Forth-Computer aufzubauen. (Mit Willem Ouwerkerk von der HCC-Forthgebruikersgroep und deren "Igel-Projekt" scheint keine Absprache zu bestehen.) Und das dritte Projekt ist das "Web Forth Project", über das Chris (der Leiter des Projekts) selbst ab Seite 35 berichtet. Forth als Java-Applet - wirklich nur zum "Ausprobieren", wegen der Langsamkeit von Java und der vielen Beschränkungen, die Java auferlegt. Forth-Tutorials mit Möglichkeiten des direkten Ausprobierens im Internet unter http://www.amsystech.com/mlosh/webforth.htm Internationale Beteiligung. Der Rezensent besorgt den deutschen Teil.

**37 Forth for the New Year**
**David Pochin**

David arbeitet auch intensiv am WebForth-Projekt mit. Hier gibt er über 4 Seiten hinweg einen Einblick darüber, wie er sich mit Forth herumschlägt. Für mich (den Rezensenten) wichtig sind seine bekennenden Eingangszeilen. Ich zitiere: "Ich verwende Forth als Amateur und ich brauche keine Unmengen von perfekten Programmzeilen zu liefern, die irgendjemand in Auftrag gegeben hat. Ich arbeite nur für mich selbst. Ich bin mein eigener Kunde. Ich verbringe gern viel Zeit damit, kleine Programmstücke zu schreiben, die genau das tun, was ich will. Keiner sieht das - und keiner interessiert sich dafür. Warum tue ich das also? Die große Flexibilität von Forth und die Schnelligkeit, mit der man unter Forth

Programme entwickeln kann, erlauben es mir, bestimmte Dinge, die mich momentan interessieren, ganz schnell zu untersuchen, ohne daß ich viel Zeit auf das Anlernen von Hintergrundwissen verwenden muß." - Soweit Dave Pochins Outing.

**41 How many Windows do humans need? Counting fruits, the classic way**
**Friederich Prinz**

Ein Artikel über OOF, mit dem sich unser aller VD-Redakteur und Direktor der FG bei den englischen Forth-Freunden vorstellt. Er geht stark auf CREATE ... DOES> ein und stützt sich auf Arbeiten von Egmont Woitzel und Michael Major. Der Artikel ist gut. Sehr erwähnenswert ist aber auch die Tatsache, daß wir (die FG) somit damit begonnen haben, Artikel zwischen der Forthwrite und der Vierten Dimension auszutauschen: In der VD erscheint (wohl schon in der vorliegenden Ausgabe) der OOF-Artikel von Chris Jakeman aus der Forthwrite 99, diesmal in deutscher Sprache. Das ist ein weiterer Schritt auf dem Weg zu einer intensivierten internationalen Zusammenarbeit. Alan Wenham, hier ein weiteres Mal lobenswert in Erscheinung tretend, hat Fritzens an sich schon sehr gutes Englisch auf Hochglanz gebracht.

**49 Letters**

Ein Brief über die Ansteuerung von Hardware per Forth, ein Brief über die FIGUK-Webseite und Forth für Anfänger und schließlich noch ein Brief mit einem flammenden Aufruf von Graeme Dunbar, einem Dozenten an der Gordon University in Aberdeen, über den Umstand, daß es keine guten Bücher (mehr) gibt, aus denen ein Anfänger Forth lernen könnte. Er ist der Meinung, es genüge nicht, wie man manchmal zu lesen bekommt, Forth-Programme von einem "Experten" so "schreiben" zu lassen, daß man sie "lesen" kann. Ein Anfänger braucht gute Kommentare in Prosa und gutgewählte Beispiele. Es genügt nicht, das Ergebnis einer langen Entwicklungs- und Überlegungsreihe dem Lernenden schließlich in knappem Programmierstil einfach nur mitzuteilen. Ein Anfänger möchte die Entstehungsgeschichte selbst mitverfolgen können und eventuelle Fehlerquellen, unter Anleitung, für sich selbst ausräumen dürfen. Ja, ist man da (der Rezensent jedenfalls) versucht zu fragen, gilt das nicht für jedes Fachgebiet? Für die Lehre schlechthin? Muß man Forth da besonders erwähnen? Ein Lehrbuch ist kein Aufsatz über jüngsterrungene Forschungsergebnisse.

# FreeMem

**- „Speichermanagement“ -**

von

**Michael Major, Friederich Prinz**

Spötter sagen, daß Windows trotz der gegenteiligen Beteuerungen seiner Fangemeinde mehr BUGs als FEATUREs habe. Diese zum Teil auf leidvoller Erfahrung beruhende‚Aussage‘ läßt sich heute so sicher nicht mehr aufrecht erhalten. Immerhin lebt aber eine ganze ‚Schattenindustrie‘ davon, mit mehr oder weniger nützlichen Tools echte und vermeintliche Fehler der diversen Windows-Varianten ‚auszubügeln‘. Ein Tool der nützlicheren Art möchten wir hier vorstellen. Zuvor sollten Sie uns aber bei einem kleinen Test begleiten.

Das Speichermanagement von Windows ist nach wie vor eine ‚hakelige‘ Angelegenheit. Dabei ist es gar nicht so, daß Windows selbst in diesem kritischen Bereich allzuviel Unsinn treibt. Vielmehr sind es in (fast) allen Fällen Fehler, Versäumnisse oder mangelnde Kenntnisse der programmierenden Zunft, die dafür verantwortlich sind, daß dem System nach einer längeren Arbeitssitzung immer wieder einmal der Speicher knapp wird. Sie können das leicht selbst überprüfen.

Wenn Sie das nächste Mal Ihr System starten, werfen Sie einmal einen Blick auf die SystemInfo. Windows wird Ihnen melden, daß es so um die 95 % der Systemressourcen frei hat. Danach sollten Sie möglichst viele, unterschiedliche Applikationen starten. Wenn Sie zwischendurch immer wieder auf die SystemInfo schauen, werden Sie sehen, daß sich diese Ressourcen recht schnell verkleinern. Je nach der Speicherausstattung Ihres Rechners werden Sie sehr schnell an einen Punkt gelangen, an dem a) Ihr Rechner extrem langsam wird, und b) die Festplatte Ihres Rechners ununterbrochen heftige Arbeitsgeräusche von sich gibt. Das System arbeitet ab diesem Zeitpunkt intensiv mit der Swap-Datei. Das RAM des Rechners ist vollständig belegt.

Und nun geben Sie alle offenen Anwendungen wieder frei. Behalten Sie dabei aber immer wieder die SystemInfo im Auge. Die Ressourcen Ihres Systems kehren allmählich zurück – allerdings nie wieder auf den Stand, den sie zu Beginn des kleinen Tests hatten !

Offenbar haben einige der von Ihnen gestarteten Applikationen den Speicher, den sie vom System zur Verfügung gestellt bekommen hatten, nicht oder nicht korrekt freigegeben. Jedenfalls sieht das System sich außer Stande, den Müll im Speicher aufzuräumen. An dieser Stelle mag man es Microsoft als BUG ankreiden, daß die Betriebssystembauer nicht im voraus wissen konnten, ob eine zukünftige Applikation xxx.yyy den allokierten Speicher gänzlich an das System zurückgeben wollte, oder ob eben diese Applikation den Speicher nur anders organisieren wollte. Jedenfalls macht es einen Unterschied, ob ein Programm einen allokierten Speicherbereich nur ‚frei‘ gibt, oder dem System mitteilt, daß es über diesen Speicher wieder nach eigenem Gutdünken verfügen, und ihn zum Beispiel aufräumen kann.

Es gibt Programme, die es ‚richtig‘ machen. Das bereits angesprochene Tool ‚*FreeMem Professional*‘ ist ein solches Programm. *FreeMem Professional* zwingt dem System eine Garbage Collection auf und räumt den Müll fort, den andere hinterlassen haben. Und das macht das Programm auf eine geradezu verblüffend einfache Weise. Sie geben dem Tool vor, wieviel freies RAM Sie auf Ihrem Rechner haben möchten. *FreeMem* allokiert diesen Speicher für Sie. Als Vordergrundtask bekommt *FreeMem* selbstverständlich das Recht, auf RAM zu zugreifen. Dem System bleibt nichts anderes übrig, als den Datenmüll in die SWAP-Datei auszulagern. Nach dem Allokieren gibt *FreeMem* den Speicher wieder frei – und teilt dem System mit, daß es ihn nicht mehr benötigt.

**Anschließend aufgerufene Programme haben wieder ein Maximum an freiem RAM zur Verfügung !**

Um den Datenmüll in der SWAP-Datei müssen Sie sich keine Gedanken machen. Das meiste davon erledigt Windows ‚mit der Zeit‘. Irgendwann wird es dem System zu viel – und es gibt auch die ‚unaufgeräumten‘ Speicherblöcke wieder zur weiteren Verwendung frei. Und zurückgelassene, das heißt, gar nicht erst frei gegebene,

Speicherblöcke sind spätestens nach dem nächsten Systemstart ebenfalls aus der SWAP-Datei verschwunden.

*FreeMem Professional* glänzt als Werkzeug mit einer Funktionalität, die gerade ausreicht, um ein wirklich gutes Werkzeug zu sein. Man kann bei seiner Bedienung gar nichts falsch machen. Selbstverständlich ist das Tool eine Windows-Applikation und ebenso selbstverständlich stellt es ‚auf Anforderung' seine Arbeit sehr anschaulich graphisch dar. Zu beziehen ist diese nützliche Shareware von dem Autor Meikel Weber, den Sie unter ***support@meikel.com*** per E-Mail erreichen können. Sie können das Programm auch direkt aus dem Internet herunterladen*:* ***www.meikel.com/freemem***.

Die beschriebene, genial einfache Vorgehensweise von FreeMem hat uns selbstverständlich dazu animiert, den ‚Trick' mit einem Forth nachzuvollziehen. Wir haben das mit dem **comForth 4** und dem **Win32For** getan:

```
\ FREEMEM - Speichertool - fep, 01/01.1999
\                        - comForth 4

CR .( lade FREEMEM Tool )
CR
   0 VALUE MemPtr
   0 VALUE MemAsk

: GetMem ( n -- )\ n bei mir : 65.000.000
Byte
    MemPtr IF DROP     \MemPtr enthält noch
                       \einen gültigen Zeiger
         ELSE DUP TO MemAsk \Speicheranforder
                            \-rung sichern
              ALLOCATE      \Speicher
                            \anfordern
              IF DROP  \System will Speicher
                       \nicht hergeben
                 ELSE TO MemPtr  \Zeiger auf
                                 \Speicher sichern
                    MemPtr MemAsk BOUNDS
                 \Schleifenparameter vorbereiten
                    DO 65 I C!
                 \angeforderten Speicher mit 'A'
                 \beschreiben
                  LOOP
              THEN
         THEN ;

: FreeMem ( -- )
    MemPtr FREE   \angeforderten Speicher
                  \freigeben
    0 TO MemPtr ; \MemPtr für nächsten
                  \Versuch rücksetzen
.( fertig )
```

Es funktioniert unter dem comForth 4, wie sich sowohl mit FreeMem, als auch durch Beobachtung der SWAP-Datei belegen läßt. Aber, wer sagt es dem Tom ? Der gleiche Code arbeitet im Win32For ohne Einschränkung – nur wird der Speicher erst dann wirklich freigegeben, wenn Win32For ebenfalls das System verläßt ! Auch das Win32For ‚ferkelt' mit dem Speicher herum. Vielleicht schauen Sie einmal nach, wie sich das von Ihnen bevorzugte Forth verhält ? Und schimpfen Sie das nächste Mal nicht sofort auf Windows...

*mkm*
*fep*

Soeren Tiedemann vervollständigt seine Werkstattarbeit zu dem F21 (VD 01/99, S. 28) mit einem Qsort Demo...

**Soeren Tiedemann**

Freiburg im Breisgau
tiedema@mail.uni-freiburg.de

Es handelt sich um einen reinen, nicht optimierten Quicksort. Diese Implementierung für den F21 belegt, compiliert im Speicher, 185 Bytes.
Es werden maximal 18-bit Integer-Werte sortiert. Die Daten können nach Beendigung des Sortierens am höchsten Bit (Bit 19) markiert sein. Dies muss vor der Weiterverwendung der Daten beachtet werden.

**Benchmark-Tests mit dem F21-Quicksort**

Als Grundlage für die Berechnung habe ich eine durchschnittliche Geschwindigkeit von 80-MIPS beim F21 für den DRAM-Bereich angenommen.

20.000 Elemente:
- Die drei schlechtesten Fälle:

1) nur gleiche Zahlen : 4.603.869.836 Instruktionen ~ 58sek
2) rückwaerts sortiert : 3.205.019.839 Instruktionen ~ 40sek
3) schon fertig sortiert : 2.803.799.844 Instruktionen ~ 35sek

Diese Fälle können aber leicht durch ein Vorprogramm gelöst werden. ( z.B.: Durchmischen vor dem eigentlichen Start )

- Zufallsverteilung:

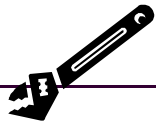
1) 19.121.860
2) 19.021.730
3) 18.635.263
4) 18.864.598 --> im Schnitt 18.774.388 Instruktionen ~ 0.24sek

60.000 Elemente:
- Zufallsverteilung:

1) 60.945.024
2) 58.808.164
3) 60.984.071
4) 59.372.072 --> im Schnitt 60.027.333 Instruktionen ~ 0.75sek

100.000 Elemente:
- Zufallsverteilung:

1) 103.457.298
2) 101.824.480
3) 101.948.066
4) 103.127.398 --> im Schnitt 102.589.310 Instruktionen ~ 1.30sek



```
\ F21-Simulator, copyright by Ultra Technology 1998,
\ rewritten for the Win32for-System by Soeren Tiedemann 1998

\ F21 -- Native Code Examples: Quicksort-Algorithm

HEX  0 ORG

MACRO >RS                  \ A: --    RS:            RP SP -2 Ret  DS: x First Last
   pop a! pop pop          \ A: Ret   RS:                     RP  DS: SP -2 F First Last
   a !r+ push push         \ A: --    RS:              RP+1 SP -2  DS: F First Last
END-MACRO

MACRO RS>                  \ A: --    RS:                RP SP -2  DS: x First Last
   pop pop over nop        \ A: --    RS:                      RP  DS: -2 SP -2 x First Last
   2/ pop nop nop          \ A: --    RS:                      --  DS: RP -1 SP -2 x First Last
   + dup a! push           \ A: RP-1  RS:                   RP-1  DS: SP -2 x First Last
   push push @a push       \ A: --    RS:          RP-1 SP -2 Ret  DS: x First Last
END-MACRO

MACRO DS2>                 \ A: --    RS:                RP SP -2  DS: x First Last
   drop pop pop over       \ A: --    RS:                      RP  DS: -2 SP -2 First Last
   + dup a! push           \ A: SP-2  RS:                 RP SP-2  DS: -2 First Last
   push drop drop nop      \ A: SP-2  RS:              RP SP-2 -2  DS: --
   @a+ push @a+ pop        \ A: --    RS:              RP SP-2 -2  DS: First Last
END-MACRO

MACRO nochmal?             \ A: --    RS:            RP SP -2 Ret  DS: First Last
   over dup dup -or        \ A: --    RS:            RP SP -2 Ret  DS: 0 Last First Last
   com 2* 2* nop           \ A: --    RS:            RP SP -2 Ret  DS: -4 Last First Last
   + com over nop          \ A: --    RS:            RP SP -2 Ret  DS: First Not(Last-4) First Last
   +                       \ A: --    RS:            RP SP -2 Ret  DS: F First Last
END-MACRO

MACRO Laeufer_hoch         \ A: up    RS:                RP SP -2  DS: x 1st up dn First Last
   begin
      drop @a+ over com    \ A: up+1  RS:                RP SP -2  DS: Not(1st) Next 1st up dn First Last
      + com @a+ drop       \ A: up+2  RS:                RP SP -2  DS: F1 1st up dn First Last
   -until
   2* com -if              \ A: up+2  RS:                RP SP -2  DS: F2 1st up dn First Last
      push push push over  \ A: up+2  RS:      RP SP -2 F2 1st up  DS: First dn First Last
      a! pop pop pop       \ A: up    RS:                RP SP -2  DS: F2 1st up dn First Last
   then
END-MACRO

MACRO Laeufer_runter       \ A: up    RS:                RP SP -2  DS: x 1st up dn First Last
   push push drop a        \ A: up    RS:          RP SP -2  x 1st  DS: up dn First Last
   over a! pop pop         \ A: dn    RS:                RP SP -2  DS: x 1st up dn First Last
   begin
      drop @a com over     \ A: dn    RS:                RP SP -2  DS: 1st Not(Next) 1st up dn First Last
      + pop dup a          \ A: dn    RS:                   RP SP  DS: dn -2 -2 F'' 1st up dn First Last
      + a! push 2*         \ A: dn-2  RS:                RP SP -2  DS: F' 1st up dn First Last
   com -until              \ A: dn    RS:                RP SP -2  DS: F 1st up dn First Last
END-MACRO

MACRO weiter?              \ A: dn    RS:                RP SP -2  DS: F 1st up dn First Last
   drop drop push drop     \ A: dn    RS:              RP SP -2 up  DS: First Last
   a pop over over         \ A: dn    RS:                RP SP -2  DS: up dn up dn First Last
   pop dup a! 2*           \ A: -2    RS:                   RP SP  DS: -4 up dn up dn First Last
   + com nop nop           \ A: -2    RS:                   RP SP  DS: Not(up-4) dn up dn First Last
   + com a push            \ A: -2    RS:                RP SP -2  DS: F up dn First Last
END-MACRO

MACRO sortieren            \ A: -2    RS:                RP SP -2  DS: F up dn First Last
   drop over a com         \ A: -2    RS:                RP SP -2  DS: 1 dn up dn First Last
   2* push over a          \ A: -2    RS:              RP SP -2 2  DS: -2 up dn up dn First Last
   + a! pop nop            \ A: up-2  RS:                RP SP -2  DS: 2 dn up dn First Last
   + dup push a            \ A: up-2  RS:           RP SP -2 dn+2  DS: up-2 dn+2 up dn First Last
   push push @a+ @a+       \ A: up    RS: RP SP -2 dn+2 up-2 dn+2  DS: [up-1,up-2] up dn First Last
   over @r+ over F0000 #   \ A: up    RS: RP SP -2 dn+2 up-2 dn+3  DS: F0000 [up-2] [dn+2] [up-2,up-1,up-2]
up dn First Last
   and -or @r+ pop         \ A: up    RS:      RP SP -2 dn+2 up-2  DS: dn+4 [dn+3,'dn+2] [up-2,up-1,up-2]
up dn First Last
   drop pop a! over        \ A: up-2  RS:           RP SP -2 dn+2  DS: ['dn+2,dn+3,dn+2] [up-2,up-1,up-2]
up dn First Last
   !a+ !a+ drop 0FFFF #    \ A: up    RS:           RP SP -2 dn+2  DS: 0FFFF [up-2,up-1,up-2] up dn First
Last
   and !r+ !r+ drop        \ A: up    RS:           RP SP -2 dn+4  DS: up dn First Last
END-MACRO
```

```
MACRO 1.tes_neu            \ A: up    RS:              RP SP -2 x  DS: up dn First Last
   pop pop dup push        \ A: up    RS:                 RP SP -2  DS: -2 x up dn First Last
   push drop push over     \ A: up    RS:           RP SP -2 -2 up  DS: First dn First Last
   a! pop pop a            \ A:'1st+2 RS:                 RP SP -2  DS: '1st+2 -2 up dn First Last
   + a! @a 0FFFF #         \ A:'1st   RS:                 RP SP -2  DS: 0FFFF 1st up dn First Last
   and over a! dup         \ A: up    RS:                 RP SP -2  DS: x 1st up dn First Last
END-MACRO

MACRO Teilen>              \ A: -2    RS:                 RP SP -2  DS: x up dn First Last
   drop drop over pop      \ A: -2    RS:                    RP SP  DS: -2 First dn First Last
   + !r+ a com             \ A: -2    RS:                  RP SP+1  DS: 1 dn First Last
   2* 2* nop nop           \ A: -2    RS:                  RP SP+1  DS: 4 dn First Last
   + dup !r+ a             \ A: -2    RS:                  RP SP+2  DS: -2 dn+4 First Last
   push push drop dup      \ A: -2    RS:          RP SP+2 -2 dn+4  DS: Last Last
   a! pop @a nop           \ A: 'nth  RS:              RP SP+2 -2  DS: nth' dn+4 Last
   0FFFF # and !a nop       \ A: --    RS:             RP SP+2 -2  DS: dn+4 Last
END-MACRO

MACRO >Teilen              \ A: --    RS:                 RP SP -2  DS: x First Last
   drop dup push over      \ A: --    RS:           RP SP -2 First  DS: Last First Last
   a! 80000 # @a over      \ A: 'nth  RS:           RP SP -2 First  DS: 80000 nth 80000 First Last
   2* 2/ 2/ -or            \ A: 'nth  RS:           RP SP -2 First  DS: nth' 80000 First Last
   !a pop a! pop           \ A: '1st  RS:                    RP SP  DS: -2 80000 First Last
   dup push over @a        \ A: '1st  RS:                 RP SP -2  DS: 1st' 80000 -2 80000 First Last
   0FFFF # and dup push    \ A: '1st  RS:             RP SP -2 1st  DS: 1st 80000 -2 80000 First Last
   -or !a+ push drop       \ A: up-1  RS:          RP SP -2 1st -2  DS: First Last
   @a+ drop over pop       \ A: up    RS:             RP SP -2 1st  DS: -2 Last First Last
   + push drop a           \ A: up    RS:          RP SP -2 1st dn  DS: First Last
   pop dup pop dup         \ A: up    RS:                 RP SP -2  DS: x 1st x dn First Last
END-MACRO

MACRO Teilen               \ A: --    RS:                 RP SP -2  DS: F First Last
   >Teilen
   begin
      Laeufer_hoch Laeufer_runter
   weiter? -while
      sortieren 1.tes_neu
   repeat
   Teilen>                 \ A: --    RS:               RP SP+2 -2  DS: First Last
END-MACRO

': Quicksort               \ A: --    RS:             RP SP -2 Ret  DS: First Last
   nochmal? -if
      >RS
         Teilen Quicksort
         DS2>   Quicksort
      RS>
   then  ;'                \ A: --    RS:                 RP SP -2  DS: x First Last

DECIMAL

\ Test- und Aufrufprogramm fuer den Quicksort

': QSrtTest
   250000 # push 500000 # push
    40100 # 100 # dup dup
    -or com 2* push
    Quicksort
    drop drop drop pop
    pop pop drop drop
    drop nop nop nop
    BREAK
;'

\ Sortierdaten fuer das Beispiel- und Aufrufprogramm
des Quicksort's

: Daten1>Speicher
   20000 0 DO
      I DUP 1+ SWAP 2* 100 + t!
   LOOP ;

: Daten2>Speicher
   20000 0 DO
      I DUP 20000 SWAP - SWAP 2 * 100 + t!
   LOOP ;

: Daten3>Speicher
   RANDOM-INIT
   20000 0 DO
      65535 RANDOM I 2 * 100 + t!
   LOOP ;
: Daten4>Speicher
   20000 0 DO
      10 I 2 * 100 + t!
   LOOP ;

Daten3>Speicher
```

# Stack-Auslagerung in eine Datei beim 32/Forth von Rick VanNorman und bei Turbo-Forth

von
Fred Behringer
Planegger Str. 24, 81241 München

Eingabedaten in Turbo-Forth auf den Stack zu bringen, ist einfach: Man bette sie in den Quelltext ein und schon sind sie drauf. In umgekehrter Richtung muß man etwas überlegen. Gesetzt den Fall, man hat sich über ein Entwicklungssystem (das sich jeder ernsthafte Forth-Programmierer über die Jahre hinweg selbst erstellt) eine Liste von Parametern berechnet, die für ein später einzulesendes Programm verwendet werden sollen. Es wird in dieser Arbeit gezeigt, wie man in Turbo-Forth den Stackinhalt in eine DOS-ASCII-Datei übertragen kann. ANS-Forth-Anpassungen werden ebenfalls diskutiert und am OS/2-Forth-System von Rick VanNorman ausprobiert.

***Stichworte: Turbo-Forth, RCVN-32/Forth, OS/2-Forth, ANS-Forth, DPMI, Windows, E/O-Datentransfer***

**Scientific Forth.** Peter Knaggs, Herausgeber des Journal of Forth Application and Research, wird in seinem Universitätsbereich schief angesehen, wenn er zugibt, sich mit Forth zu beschäftigen ([2] über [3]). Mir sind solche Situationen nicht ganz unbekannt.

**Numerical Recipes.** Seit Julian V. Nobles 1991 erschienenem Buch "Scientific Forth" [5] sind Bestrebungen im Gange, Forth zu einer für das wissenschaftliche Rechnen akzeptablen Sprache auszubauen. Auszubauen. Es ist ja schon alles da. Es kann ja jeder ohne Schwierigkeiten der Bauart "Das geht nicht und jenes geht nicht" sofort alles selbst machen. Aber eben drum. Für manche allzusehr Fachbezogene, die den Computer nur als fertiges Werkzeug des Rechnens betrachten wollen, muß das erst gemacht werden. Sie wollen sich selbst damit nicht aufhalten. Seit 1994 wird auf dem Taygeta-FTP-Server unter der Federführung von eben Julian V. Noble eine "Scientific Forth Library" hochgezogen. - Man müßte diese Bestrebungen mehr bekannt machen und sich dafür einsetzen. Ein Buch "Numerical Recipes in Forth" hat wohl neben C, Pascal und FORTRAN noch keiner in Erwägung gezogen?

**Wissenschaftliches Rechnen** spielt sich zu einem großen Teil interaktiv ab. Man hat eine Idee, man entwickelt ein Verfahren, man gießt es in einen Algorithmus und man implementiert diesen in irgendeiner Sprache, die man leidlich beherrscht und die (nach gängiger Meinung) für den angestrebten Zweck geeignet ist. Das Ganze gelingt nie im ersten Guß. Im Gegenteil, in Abhängigkeit von den Ergebnissen des allerersten Rechengangs kommen einem erst die richtigen Ideen, wie man es eigentlich hätte machen müssen, man überlegt und man versucht es erneut. Bei diesem Versuch-und-Irrtum-Prozeß kommt es zwangsläufig immer wieder vor, daß man Ergebnisse eines vorausgegangenen Rechengangs in einem nachfolgenden als Eingabedaten (direkt oder mit leichten Änderungen) verwenden möchte. Erneutes Eintippen gilt nicht. Das wäre eine menschenunwürdige Tätigkeit.

**Stack <=> Datei.** Der normale Ort, an welchem Ausgabedaten am PC oder am Arbeitsplatzrechner aufbewahrt werden, ist die Datei (auf Diskette, Festplatte, ZIP-Medium oder was auch immer als Laufwerk angesprochen werden kann). In Forth ist der normale Zwischenspeicher für Ein- und Ausgabedaten der Stack. Es liegt also nahe, nach einer Routine zu suchen, die den Stackinhalt oder einen näher zu kennzeichnenden Teil davon in eine Datei "auszulagern" gestattet. Am besten gleich in einer solchen Form, wie man sie auch von Hand aus in den Quelltext eingeben würde, also in ASCII-Form. Ich will nicht alles auf einmal anstreben wollen. Auf das Prinzip kommt es an. Ich beschränke mich daher auf vorzeichenbehaftete Ganzzahlen, also solche, deren Umwandlung über **#S** bewerkstelligt werden kann. - Für den umgekehrten Vorgang, Zahlen in ASCII-Form aus einer Datei auf den Stack zu laden, ist natürlich in jedem Forth-System schon eine Möglichkeit vorgesehen. Das ist ja dann einfach nur das, was man unter "Quelldatei" versteht, vom "Interpreter" interpretiert. In den hier zu besprechenden beiden Systemen geht das über **INCLUDE <name.ext>**.

**Eigentlicher Beweggrund.** Boolesche Funktionen mehrerer Veränderlicher können als Disjunktive Normalform (DNF) dargestellt werden (z.B. [1]). Ordnet man jeder in einer bestimmten DNF enthaltenen Vollkonjunktion (Konjunktion aller beteiligten Booleschen Variablen in affirmativer oder negierter Form) eine 1 zu und den nicht enthaltenen eine 0, dann erhält man in Binärdarstellung eine Index-Zahl, die die betreffende DNF und damit die dahinterstehende Boolesche Funktion eindeutig charakterisiert. Hängt die Boolesche Funktion von n Variablen ab, so ist die Index-Zahl (bis zu) $2^n$ Bit lang. Solche Index-Zahlen können in einem Bitvektor abgespeichert werden. Ein solcher Bitvektor ist als Vektor-Konstante aufzufassen: Verschiedene Bitvektoren der Länge n stellen in eindeutiger Weise verschiedene Boolesche Funktionen von n Veränderlichen dar. Zur Erzeugung der Bitvektoren (Variablenbelegung für Variablenbelegung interaktiv an der Tastatur durch Eingabe von 1 oder 0) kann man natürlich ebenfalls ein Forth-Programm verwenden. Ein solches Programm kann dem Editor, der zur Herstellung des Programm-Quelltextes verwendet wird, zugeordnet werden. Anders ausgedrückt: Man braucht eine Möglichkeit, das Ergebnis einer Bitvektorerzeugungssitzung für den Moment festzuhalten und nach dem Compilieren des eigentlichen Programms als Datensatz wieder einzulesen. Für das zwischenzeitliche "Auslagern" bietet sich das Abspeichern in eine Datei an.

**OS/2-Forth.** Der Autor von Turbo-Forth, Marc Petremann aus Frankreich, brauchte im Titel nicht genannt zu werden: Turbo-Forth 2.1.0 [6] war eher da (1987) als das vergleichbare F-PC (1988) von Tom Zimmer [8]. Daß Turbo-Forth keine Verbreitung gefunden hat, liegt sicher nicht im System selbst begründet. Das 32-Bit-Forth von Rick VanNorman [7], das es

seit 1993 gibt, findet auch kaum Beachtung, ist aber aus mehreren Gründen äußerst bemerkenswert: Es arbeitet unter DOS **und** OS/2 im Protected-Mode, verwendet DPMI, um die volle Interrupt-Anbindung zu gewährleisten, kann als abermalige Erweiterung (schon Turbo-Forth ist eine solche) von F83 von Laxen and Perry [4] angesehen werden, verwendet insbesondere die überaus offene Metacompilier-Technik (es braucht kein C++-Compiler angeworfen und keine andere forthfremde Sprache (C++) beherrscht zu werden) - und ist ein komplettes ANS-Forth-System. Für die DPMI-Anbindung werden eigene Mittel bereitgestellt, die einfachste Arbeitsumgebung ist aber die, Windows 3.11 aufzurufen und in die DOS-Box (Großbildschirm einstellen!) zu gehen. DPMI steht dann automatisch zur Verfügung und alles, was zu tun übrigbleibt, ist, das (für DOS bestimmte) Hauptprogramm FORTH.COM aufzurufen. Ein weiterer Vorteil besteht übrigens darin, daß man sich das System von Rick VanNorman vom (amerikanischen) ftp-Server Taygeta kostenlos herunterladen kann. PKUNZIP OS2FORTH.ZIP T: -d stellt den vollständigen Verzeichnisbaum ins Laufwerk (T: ist bei mir die RAM-Disk).

**Übertragung von DOS-Datei zum Datenstack.** Die Übertragung erfolgt (in Turbo-Forth von Marc Petremann wie in 32/Forth von Rick VanNorman) per **INCLUDE <datei.ext>**. Die auf den Stack einzulesenden Daten müssen natürlich in ASCII-Form auf der Datei liegen. Die nicht ganz so einfache Frage ist die umgekehrte: Wie kommen die Daten in geeigneter Form vom Stack in die (eine) Datei? Ich betrachte diese Frage im nächsten Abschnitt. Als Datei-Erweiterung wird in Turbo-Forth **.FTH** erwartet, was aber beim Einlesen per **INCLUDE** auch weggelassen werden kann. In 32/Forth wird **.4** erwartet. Ein Weglassen ist nicht möglich und wird mit einer Nachfrage geahndet.

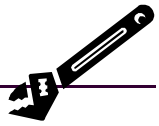
**Übertragung vom Datenstack zur DOS-Datei.** Das ist der eigentliche Gegenstand der vorliegenden Arbeit. Ich habe ein Programm mit ein und derselben Funktion einmal für Turbo-Forth (**SAVESTAK.FTH**) und einmal für 32/Forth (**SAVESTAK.4**) verfaßt. Syntax für den Aufruf zur Abspeicherung des Stacks ist beispielsweise **SAVE-STACK WWW.FTH** für Turbo-Forth und **SAVE-STACK WWW.4** für 32/Forth. Die Dateinamen können aber natürlich auch weitgehend beliebig anders gewählt werden. **WWW.FTH** (**WWW.4**) ist die Datei, in welche die Stackdaten geschrieben werden. Ist sie noch nicht vorhanden, wird sie angelegt. Auf dem Stack müssen genügend Daten liegen. Wenn nicht, erscheint eine Fehlermeldung. Zuoberst auf dem Stack muß die Anzahl der zu übertragenden Stackeinträge liegen. Das Programm ist gegen eine zu große (größer als momentane Stacktiefe) und eine zu kleine (0 oder negativ) Angabe abgesichert (Fehlermeldung und Aussprung).

**Umwandlung Stack - ASCII.** Der Forth-Stack wirkt als LIFO (last in - first out). Ich hätte die Daten zur Weiterverarbeitung gern in FIFO-Form (first in - first out). Zur Umkehrung verwende ich (sicher nicht die schnellste Art, aber vom Konzept her einfach) einen Zwischenspeicher. In 32/Forth nehme ich dazu den 128 KB langen Puffer **EDBUF** des Editors mit **END** als Zeiger, in Turbo-Forth nehme ich dafür den freien Dictionary-Platz zwischen **HERE** und **SP@**. Die Stackdaten werden nacheinander abgerufen und in den Zwischenspeicher gelegt (das oberste Stackelement natürlich zuerst) und danach vom Zwischenspeicher nacheinander wieder hereingeholt und auf den Stack gelegt (das ursprünglich oberste Stackelement jetzt wieder als erstes, also diesmal zuunterst). Danach werden die Daten Eintrag für Eintrag vom Stack geholt, per **<# #S #>** in ASCII-Form umgewandelt und byteweise in den (wieder denselben) Zwischenspeicher gelegt. Wird dabei (unbeabsichtigterweise) die Kapazität des Zwischenspeichers überschritten, so erscheint eine Fehlermeldung und der Vorgang wird abgebrochen.

**Bewahrung der Basis.** Im Programm (siehe Listing) habe ich Wert darauf gelegt, es so einzurichten, daß die Basis nach dem **INCLUDE**n wieder genauso aussieht wie vorher. Die abzuspeichernde Datei mit den zu übernehmenden Stackdaten wird (als ASCII-Text) so angelegt, daß einige Angaben zur Basis übernommen werden: Beim späteren Einlesen der dateigespeicherten Stackdaten wird zunächst die momentane Basis zwischengespeichert, dann wird die Basis auf den Wert geschaltet, unter welchem die Stackdaten (in ASCII-Form) ursprünglich abgespeichert wurden, und am Ende, nach dem Einlesen der Stackdaten, wird die Basis wieder auf den Wert vor Beginn des Einlesens zurückgeschaltet. Wurden die Stackdaten beispielsweise dezimal in die Datei gespeichert, ist aber das System beim **INCLUDE**n der Datendatei auf **HEX** geschaltet, so bleibt sichergestellt, daß die Daten in ihrer ursprünglichen Bedeutung (dezimal) übernommen werden, das System danach aber wieder auf **HEX** eingestellt erscheint.

**End-of-file.** Mein bevorzugtes Textverarbeitungssystem (Context-Pro) schließt, als Editor eingesetzt, ASCII-Text-Dateien grundsätzlich mit einem <EOF> (ASCII-Wert 1A) ab. Hole ich solchermaßen erstellte Quelltexte in Turbo-Forth per **INCLUDE <datei.ext>** in das System, so stört das nicht: Das EOF-Zeichen wird beim Einlesen ignoriert.

**Ausweg.** Im 32/Forth-System von Rick VanNorman erzeugt das Einlesen eines Quelltextes mit abschließendem EOF-Zeichen per **INCLUDE <datei.ext>** eine Fehlermeldung: Das System versucht, den ASCII-Code 1A zu "interpretieren", was natürlich ohne Erfolg bleibt. Man könnte mit dieser Fehlermeldung leben, sie ist aber unschön. In Turbo-Forth kann man durch Einsetzen der Buchstabenkombination **EOF** in den Quelltext diesen für das Einlesen per **INCLUDE** an beliebiger Stelle "abschneiden". Der Rest wird dann vom Interpreter ignoriert und kann für Erklärungen und Anmerkungen genutzt werden. Ein solches **EOF** scheint es in Rick VanNormans 32/Forth nicht zu geben. Es gibt aber einen Ausweg: Die öffnende runde Klammer. Die runde Klammer als ANS-Forth-Wort ist in Rick VanNormans System so komfortabel definiert, daß sie bis zum Ende einer einzulesenden Textdatei wirkt und alles nach ihr Kommende unberücksichtigt läßt.



**In Turbo-Forth** läßt die Wirkung der Klammer überraschenderweise schon nach einer einzigen **TIB**-Füllung ("logische Zeile") nach : Das Ende der Zeile wirkt wie eine schließende Klammer. Die öffnende Klammer muß dann, wenn man das unbedingt so haben möchte, jeder logischen Zeile neu vorangestellt werden.

**ANS-Forth.** Die Erklärungen der ANS-Forth-Dokumentationen sind diesbezüglich verschwommen: Die öffnende Klammer soll danach bis zur nächsten schließenden Klammer (bis hierher noch selbstverständlich) oder aber bis "zum Ende des durchzukämmenden (des zu parsenden) Programmteils" wirken. Was heißt hier "Ende"? In Turbo-Forth wird das nächste "Ende" eben von der Aufnahmefähigkeit des Terminal-Input-Buffers (**TIB**) diktiert.

**Backslash.** Der "umgekehrte" Schrägstrich schafft es übrigens in Rick VanNormans System auch nicht, das **EOF**-Zeichen 1A (das von Context-Pro ganz zum Schluß an den Anfang einer neuen Zeile gestellt wird) für den Interpreter unsichtbar zu machen.

**Programm-Listing für Turbo-Forth von Marc Petremann**

```
BASE @
HEX

: CELL  ( -- 2 )     2  ;              \ ANS-Anpassung, nicht in Turbo-Forth

VARIABLE BEG-BUF                       \ Übertragungspuffer
VARIABLE END-BUF                       \ Ende von diesem
VARIABLE PTR-BUF                       \ Zeiger zu diesem

: ERR-BUF ( -- )                       \ Puffer-Grenzen überschritten?
     PTR-BUF @ 100 + END-BUF @ U>      \ Man beachte, daß > hier
     ABORT" Datensatz zu lang!" ;      \ unangebracht wäre!

: CRLF>BUF ( -- )                      \ CRLF in Übertragungspuffer einfügen
     0D PTR-BUF @    C!                \ CR
     0A PTR-BUF @ 1+ C!                \ LF
     2 PTR-BUF +! ERR-BUF ;            \ Zeiger PTR weiterrücken, Fehlertest

: $>BUF ( ad cnt -- )                  \ String bei ad
     >R PTR-BUF @ R@ CMOVE             \ in Übertragungspuffer legen.
     R> PTR-BUF +! ERR-BUF ;           \ Zeiger PTR weiterrücken, Fehlertest

: S>BUF ( cnt -- )                     \ cnt Stackeinträge
     0                                 \ in Übertragungspuffer legen.
     DO                                \ Stackeintrag für Stackeintrag
       S>D <# #S #> >R                 \ nach ASCII umwandeln und
       PTR-BUF @ R@ CMOVE              \ in Übertragungspuffer legen.
       R> PTR-BUF +! CRLF>BUF          \ Zeiger anpassen und CRLF hinzufügen.
     LOOP ;

: SAVE-STACK ( <name> n1 ... nm m -- ) \ m Daten von Stack in Datei speichern
     HERE 200 + DUP BEG-BUF !          \ Übertragungspuffer
     PTR-BUF ! SP@ 200 - END-BUF !     \ Zeiger und Ende initialisieren
     DUP 0> NOT                        \ Absicherung gegen zu kleine
     ABORT" Datenzahl muß > 0 und < 32768 sein!"
     DEPTH OVER <=                     \ und zu große Eingaben.
     ABORT" Zu wenig Daten auf dem Stack!"
     " BASE @ HEX "   $>BUF            \ Basis zur Zeit der Abspeicherung
     BASE @ DUP HEX 1 S>BUF BASE !     \ in Datei beim Einlesen der
     " BASE ! "       $>BUF            \ Datei wiederherstellen.
                   CRLF>BUF            \ Mit CRLF abschließen
     PTR-BUF @ >R >R R@ 0              \ Anzahl Stack-Einträge
     DO
       PTR-BUF @ !
       CELL PTR-BUF +! ERR-BUF
     LOOP                              \ Stack auslagern und in
     R> R@ PTR-BUF ! >R R@ 0           \ umgekehrter Reihenfolge
     DO                                \ wieder hereinholen.
       PTR-BUF @ @
       CELL PTR-BUF +! ERR-BUF
     LOOP
     R> R> PTR-BUF ! >R                \ Zeiger zurücksetzen
     R@               S>BUF            \ m Stackeinträge übertragen
     R> 1             S>BUF            \ Die Basis, die zuunterst liegt,
     " ROLL BASE ! "  $>BUF            \ hervorholen und wiedereinstellen
     BEG-BUF @ PTR-BUF @ SAVE ;        \ Puffer nach Datei übertragen
```

```
BASE !                                  \ Basis wie vor INCLUDEn dieser Datei

EOF
```

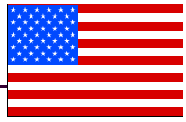
Programm-Listing für Rick VanNormans 32/Forth 3.05

```
BASE @
HEX
ALSO EDITOR                             \ Damit man an den Editor-Puffer kommt

: EDBUF-ERR ( -- )                      \ Puffer-Grenzen überschritten?
     END 300 + EBUF >
     ABORT" Datensatz zu lang!" ;

: CRLF>EDBUF ( -- )                     \ CRLF in Editor-Puffer einfügen
     0D END    C!                       \ CR
     0A END 1+ C!                       \ LF
     END  2 + TO END EDBUF-ERR ;        \ Zeiger END weiterrücken, Fehlertest

: $>EDBUF ( ad cnt -- )                 \ String bei ad
     >R END R@ CMOVE                    \ in Editor-Puffer legen.
     END R> + TO END EDBUF-ERR ;        \ Zeiger END weiterrücken, Fehlertest

: S>EDBUF ( cnt -- )                    \ cnt Stackeinträge
     0                                  \ in Editor-Puffer legen.
     DO                                 \ Stackeintrag für Stackeintrag
       S>D <# #S #> >R                  \ nach ASCII umwandeln und
       END R@ CMOVE                     \ in Editor-Puffer legen.
       END R> + TO END CRLF>EDBUF       \ Zeiger anpassen und CRLF hinzufügen.
     LOOP ;

: SAVE-STACK ( <name> n1 ... nm m -- ) \ m Daten von Stack in Datei speichern
     DUP 0> NOT                         \ Absicherung gegen zu kleine
     ABORT" Datenzahl muß > 0 und < 80000000h sein!"
     DEPTH OVER <=                      \ und zu große Eingaben.
     ABORT" Zu wenig Daten auf dem Stack!"
     INITIALIZE BUF TO END >R           \ Initialisierung, END als Zeiger
     " BASE @ HEX "   $>EDBUF           \ Basis zur Zeit der Abspeicherung
     BASE @ DUP HEX 1 S>EDBUF BASE !    \ in Datei beim Einlesen der
     " BASE ! "       $>EDBUF           \ Datei wiederherstellen.
                   CRLF>EDBUF           \ Mit CRLF abschließen
     END R@ 0                           \ Anzahl Stack-Einträge
     DO     TUCK ! CELL+ LOOP DROP      \ Stack auslagern und in
     END R@ 0                           \ umgekehrter Reihenfolge
     DO DUP @ SWAP CELL+ LOOP DROP      \ wieder hereinholen.
     R@               S>EDBUF           \ m Stackeinträge übertragen
     R> 1             S>EDBUF           \ Die Basis, sie liegt zuunterst,
     " ROLL BASE ! "  $>EDBUF           \ hervorholen und wiedereinstellen
     INPUT-NAME FILENAME OUTPUT-FILE    \ Editor-Puffer nach Datei übertragen
     ;

PREVIOUS                                \ EDITOR wieder aus dem CONTEXT nehmen
BASE !                                  \ Basis wie vor dem INCLUDEn dieser Datei

: EOF ( -- )                            \ Hommage á Marc Petremann
     [COMPILE] ( ; IMMEDIATE            \ und sein Turbo-Forth

EOF                                     \ Und schon wird EOF angewendet
```


[1] Böhme, G.: Einstieg in die Mathematische Logik, Hanser-Verlag München 1981.
[2] Deliano, R.: Leserbrief über Peter Knaggs´ Artikel, Vierte Dimension 12 (1996), Heft 3, S.6.
[3] Knaggs, P.: Forth auf eine professionelle Basis heben, Vierte Dimension 12 (1996), Heft 1, S.19-20.
[4] Laxen, H., and Perry, M.: F83, 16-Bit-Forth, blockorientiert, 1983.
[5] Noble, J.V.: Scientific Forth, Mechum Banks Publishing 1992.
[6] Petremann, M.: Turbo-Forth, hier verwendete Version aus dem Jahre 1989 noch 16-Bit-Version, dateiorientiert, deutsche Bearbeitung von Fred Behringer.
[7] VanNorman, R: "32-bit Protected-Mode Subroutine Threaded Forth" für DOS und OS/2, dateiorientiert, 1993.
[8] Zimmer, T.: F-PC, 16-Bit-Forth, 1988.


Den Gesamtartikel, aus dem man das Programm leicht herausziehen kann, schicke ich auf Anfrage gern elektronisch zu. Turbo-Forth in deutscher Version könnte ich ebenfalls beilegen.


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



Lieber Friederich,

am 14. November 1989, veranstaltete die Silicon Valley Forth Interest Group ihren jährlichen Forth-Tag. Seine Besonderheit besteht darin, daß er jedermann von fern und nah anzieht, der etwas über Forth hören, erzählen, zeigen oder erfahren möchte. Solange ich mich erinnern kann, fand der Forth-Tag immer eine oder zwei Wochen vor der FORML-Konferenz in Asimolar statt, Chuck Moore war da, um seine Ansichten mit uns zu teilen, und Dr. Ting sorgte nicht nur für intellektuelle, sondern auch für irdische Genüsse (in Form eines herrlichen Grillessens zur Mittagszeit).
Ich muß sagen, daß ich angenehm überrascht war, schon 10 Leute im Raum vorzufinden, als ich kurz vor 10 Uhr eintraf. Und die Zahl wuchs auf mehr als 30, während die Vorträge liefen. Der Forth-Tag ist wie ein Schul- oder Familientreffen; es ist erstaunlich wie viele lang verloren geglaubte 'Verwandte' auftauchen, und sei es nur um den anderen zu zeigen, daß es sie noch gibt.
Dr. Ting eröffnete das Treffen mit einer Demonstration, die einen Laser-Pointer nutzte, der mit Hilfe eines 2-Achsen-Kopfes - bewegt von zwei Schrittmotoren, die ihre Befehle von eForth erhielten - an die Wand 'schrieb'. Die Motoren waren zwar für das eigentliche Projekt nicht schnell genug, aber die Lektion bestand nach Dr. Ting darin, daß es genügt, die Worte READ, LOAD und SEND zu eForth hinzuzufügen, um eine komplette Programmierumgebung für embedded Systeme unter Nutzung vom Windows Hyperterminal und Textfiles zu bilden.
Als nächstes kam der hervorragende Meister von Forth, Wil Baden, von weit her angereist - auf seinem Weg zur FORML - um über SOOP (simplified object oriented programming - vereinfachte objektorientierte Programmierung) für Forth zu lesen. SOOP, sagte er, ist eine Erweiterung von CREATE-DOES>. Es vereinheitlicht und vereinfacht Applikations-Vokabulare. Ich bin nicht kompetent genug, um Euch eine gute Zusammenfassung von Wil's Rede zu geben, aber er schrieb seine Adresse an die Tafel: WILBADEN@NETCOM.COM, damit interessierte Teilnehmer direkt von ihm lernen können.
Jon Rible kam als nächster, um sein neuestes Design eines Hybrid-Stack RISC-Mikroprozessors vorzustellen; den QS4, realisiert mit FPGA's auf einem XILINX Kit. Das ist das vierte von John's "Quicksand"-Projekten, wobei er das meiste davon, soweit ich verstanden habe, in den Kursen zum Hardware-Entwurf entwickelt hat, die er unterrichtet. Der eingebaute Interpreter basiert natürlich auf Forth. Nebenbei sagte uns John, der eines der aktivsten Mitglieder der ANSForth Bestrebung ist, das ein Update von ANSForth auf dem Web liegt (von Greg Baley, wenn ich richtig gehört habe); schaut es Euch an unter ***ftp://ftp.minerva.com/pub/X3j14/j14-cur.htm***.
David Jaffe, der Entwickler von RALPH, der "fingerspelling" (Gebärdensprache?) Hand, sprach über ein neues Gerät für Behinderte und ältere Personen, an dem er arbeitet. "Tragbare Datenerfassungs-Systeme" ("Wearable data acquisition systems" - WDAS) werden entwickelt, um zu studieren, wie sich verschiedene Arbeiten oder Bedingungen auf die Fähigkeit zur Wahrung des Gleichgewichtes auswirken. Die tragbaren Einheiten sind kleine 1/4" große Würfel, die 3-Achsen-Beschleunigungsmesser enthalten und an den Brillengläsern oder an der Hüfte befestigt werden.
Forth wird sowohl in den tragbaren Geräten, als auch in den Bedieneinheiten verwendet. Wer weiß, vielleicht bringt uns die Zukunft aufblasbare Bekleidung, die durch WDAS-Einheiten ausgelöst werden.
Skip Carter, unser Präsident, war wieder da, um Diskussionen über die wahrhaft neuen und einzigartigen Dinge, die gemeinsam so wesentlich für Forth sind, zu führen. Es scheint mir, daß er ein sechstes zu den fünf hinzugefügt hat, über die ich Euch letzten August schrieb. Ich kann einfach die neue Liste hinschreiben...
Forth wird benutzt als:
1. ein Weg, über das Denken nachzudenken
2. ein Weg, über das Berechnen nachzudenken
3. ein Weg, über das Design von Computern nachzudenken
4. ein Weg, über das Design von Programmen nachzudenken
5. eine allgemeine Universal-Programmiersprache
6. ein Weg, Hardware anzusteuern.

Als Beispiel für die Denker, für die ersten Punkte, nannte er Papert, Minsky, Moore und Wirth, in dieser Reihenfolge. Die Schwäche im Erkenntnisprozeß bei der Computerarbeit, sagt Skip, stammt aus dem Überspringen von Grenzen ohne dabei die Art des Denkens umzuschalten.
Mikhail Montvelishsky (Ihre habt vielleicht Fred Behringer's Übersetzung eines seiner Artikel in der VD 4/1995 gelesen) war mit einem Muster einer funktionierenden TV/Netz Box von iTv Corporation da. Eine Menge Leistung in einem kleinen Gehäuse, wie es scheint, fertig für den Markt, zu einem attraktiven Preis.
Nach Dr. Ting's Grill-Mittagessen hatte die Gruppe nicht nur im Gewicht, sondern auch in der Anzahl zugenommen. Ich zählte 35.
Jeff Fox setzte das Treffen mit einem Vortrag über Chuck Moore's F21Emulator und Simulator fort. Ihr könnt mehr Informationen im Web auf ***www.ultratechnology.com*** finden.
Leonard Morgenstern, leider nicht in der Lage teilzunehmen, hatte unseren Tagungsleiter, George Perry, gebeten, seinen Vortrag zu halten. Darin ging es um Methoden für CRC-Prüfsummen, die von Gordon Charlton aus UK geschrieben waren. Es war gut, daß Wil Baden da war, um George's geschickten Vortrag zu ergänzen. Unglücklicherweise, habe ich scheinbar keine umfassenden Notizen gemacht. So laßt uns zum Höhepunkt des Tages kommen---Chuck Moore kam gegen 3 Uhr nachmittags mit einem seiner üblichen farbigen Vorträge, dieses Mal um so mehr, da er über seine neue Version seines COLOR FORTH sprach. Ich muß zugeben, daß ich gemischte Gefühle habe, darüber auf meinem XT zu berichten, der nur einen monochromen Hercules Monitor hat, aber ich vermute, daß die Verwendung von Farbe Teil des Fortschritts ist, und kein Abenteuer zu mehr Komplexität. Wie auch immer, COLOR FORTH kommt mit einem eingebauten Dia-Generator, seine Screen Dumps auf einem HP

Inkjet Drucker. Seine Kommandozeile am unteren Bildrand scrollt von rechts, um Pixel auf dem Screen zu sparen (obwohl sich wohl niemand darum echte Sorgen macht). Funktionstasten vor Wörtern werden genutzt, um die Farben zu beschreiben. Blau ist Kommentar, definierte Worte sind Rot. Der Rumpf der Definition ist Grün. Schwarze Worte werden ausgeführt. Das Ende einer Definition wird durch das Fehlen grüner Worte markiert.
Damit ist COLOR FORTH ein anderes Werkzeug von einem der am meisten originellen Werkzeugmacher in der Computerindustrie. Chuck nutzt es für seinen Chip-Entwurf, und auch darüber sprach er ausführlich. Aber ich höre wahrscheinlich besser hier auf, der Brief wird wird langsam zu lang. (Ich verspreche der Nächste wird kürzer.)
Bis dann,

*--Henry*

—— übersetzt von Thomas Beierlein

Lieber Friederich,

Dein Deutsch klingt soviel besser als mein Englisch, selbst wenn Du in Eile schreibst. Ich bedauere, daß Du mehr zu tun hast, als Dir lieb ist, aber daß muß Dich doch jung halten, 'nicht Wahr' ?
Es ist das gleiche bei mir, wenn ich nach dem SVFIG-Treffen einige Tage vergehen lasse, dann fällt es mir schwer, mich hinzusetzen und einen Bericht darüber zu schreiben.
Ich verpaßte die erste Stunde des Februar-Treffens, so bekam ich nur die Hälfte von Dr. Ting's Rede über seine Ideen und Bemühungen mit, einen voll integrierten Computer auf einem Chip zu entwickeln. Er möchte ihn SOC nennen, das steht für "System On a Chip"; das Betriebssystem ist natürlich Forth, der Rest besteht aus CPU, RAM,l I/O usw. Alles was benötigt wird, ist ein Interesse des Marktes und Risiko-Kapital, mehr als man von den 16 oder 18 Leuten in unserem Publikum erwarten kann.
Dave Jaffe war über die Notwendigkeit zur Berechnung trigonometrischer Funktionen in embedded Applikationen gestolpert und hat demzufolge seine eigenen Forth-Programme entwickelt, die dies unter Nutzung von Taylor- und MacLaurin-Reihen mit der notwendigen Genauigkeit berechnen. Er zeigte uns seine Technik für die Bestimmung des Arkustangens.
Dwight Elvey, unser Bibliothekar (der zu den ersten Leuten gehört, die das 1979er fig-Forth auf einem 8080 installierten) hatte seine eigene funktionsfähige Intel 4004 Box mitgebracht, die, obwohl Intel sie 1971 als "computer on a Chip" beworben hatte, beträchtlich größer als der Chip war, den Dr. Ting heute mit seinem SOC im Sinn hat. Dwight und Dave erheitern uns recht oft mit ihrer Sammlung von "High-Tech" Überbleibseln aus den Tagen als wir noch "jung" waren.
Kevin Appert zeigte eine Serie von Bildern von einer seiner kürzlichen Dienstreisen. Sie waren auf einer Floppy Disc, entwickelt von Seatlle Filmworks, und wir waren in der Lage, sie dank Dwight Elvey's Laptop und dem Projektor des Cogswell College (welches unser Heim für die Treffen ist) zu betrachten. Kevin hatte auch eine Menge interessanter Web-Seiten am schwarzen Brett ausgehängt (wobei 'schwarzes Brett' jetzt ein irreführender Begriff ist, da es weiß ist und man mit einem Tinten-Stift anstelle von Kreide darauf schreiben muß).
Zum Abschluß des Tages verbrachten wir noch einige Zeit mit der Diskussion, wen wir in den zukünftigen Treffen einladen wollen und worüber sie sprechen sollen... Aber hier war nicht allzuviel Enthusiasmus dabei. Es scheint mir, daß die Gruppe völlig glücklich ist, einen Platz zu haben, zu dem man einmal im Monat entfliehen kann, in einer gemeinsamen Sprache zu plaudern, Informationen auszutauschen, zu lachen, zu lernen und das zu genießen, was immer auch uns wieder hierher zurückkehren läßt.
Und so geht es weiter. Bis zum nächsten Mal. Grüße an Euch alle!

Henry.

## Forth-Gruppen regional

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

**Mannheim**
Thomas Prinz
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-8632 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

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

## µP-Controller Verleih

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

## Gruppengründungen, Kontakte

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

## Forth-Hilfe für Ratsuchende

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

Jörg Staben
Tel.: 02103-24 06 09 (p)

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

## Spezielle Fachgebiete

| | |
|---|---|
| Arbeitsgruppe MARC4 | Rafael Deliano<br>Tel./Fax: 089-841 83 17 (p) |
| FORTHchips<br>(FRP 1600, RTX, Novix) | Klaus Schleisiek-Kern<br>Tel.: 040-375 008 03 (g) |
| F-PC & TCOM, Asyst<br>(Meßtechnik), embedded<br>Controller (H8/5xx//<br>TDS2020,TDS8092<br>Fuzzy | Arndt Klingelnberg, Consultants<br>akg@aachen.forth-ev.de<br>Tel.: ++32 +87 - 63 09 89<br>Fax: ++32 +87 - 63 09 88 |
| KI, Object Oriented Forth,<br>Sicherheitskritische<br>Systeme | Ulrich Hoffmann<br>Tel.: 04351 -712 217 (p)<br>Fax: -712 216 |
| Forth-Vertrieb | volksFORTH / ultraFORTH<br>RTX / FG / Super8 / KK-FORTH<br>Ingenieurbüro Klaus Kohl<br>Tel.: 08233-3 05 24 (p)<br>Fax : 08233-99 71<br>mailorder@forth-ev.de |
| Forth-Mailbox (KBBS) | 0431-533 98 98 (8 N 1)<br>Sysop Holger Petersen<br>hp@kbbs.org<br>Tel.: 0431-533 98 96 (p) bis 22:00<br>Fax : 0431-533 98 97<br>Helsinkistraße 52<br>24109 Kiel |

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

Schreiben Sie einfach der VD - oder rufen Sie

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.


**FORTH-Tagung'99**

**vom 23. bis 25.April 1999 in Oberammergau**

Liebe FORTH-Freunde,

die FORTH-Tagung'99 findet in der Zeit vom 23. bis 25. April in Oberammergau statt, aber das dürfte wohl bekannt sein.

Sehr gut angenommen wurde der von uns angebotene zusätzliche Tag, am 22. April, also einen Tag vor der Tagung, für eine Bergwanderung oder eine Schlossbesichtigung. Diese Aktivitäten bieten wir, je nach Witterung und den Wünschen der Teilnehmer, für den Donnerstag nachmittag und für den Freitag vormittag an.
Oberammergau und die Sehenswürdigkeiten ganz in der Nähe möchten wir Ihnen etwas näher bringen.
Zu einem Besuch laden ein: *Oberammergaus Heimatmuseum* mit der sehr bekannten *Krippensammlung und Holzschnitzereien* und das *Pilatushaus* mit seiner *Lüftlmalerei*.
Die schönsten *Schlösser von König Ludwig II., Linderhof und Neuschwanstein* sind von Oberammergau gut zu erreichen und die *Benediktinerabtei Ettal mit Brauerei und Likör-Fabrik* können wir besichtigen.

*Für Gipfelstürmer bietet sich die Laber-Bergbahn an,* oder Sie bemühen sich per pedes zum Kofel, der von seinem Gipfel einen herrlichen Blick über Oberammergau und das Ammertal bietet. Entspannen und erholen kann man sich anschließend im *Freizeitzentrum Wellenberg Oberammergau.*

Als Tagungshotel haben wir das Parkhotel Sonnenhof in Oberammergau ausgewählt. Alle Zimmer sind mit Bad/Dusche, WC, Telefon, Radio, Kabel-TV und Balkon ausgestattet.

Die Anreise mit dem Auto oder der Bahn über München nach Oberammergau ist bequem möglich.

Ihre Anmeldung sollten Sie sobald wie möglich an das Tagungsbüro schicken, da wir das Zimmerkontingent im Hotel bereits in der ersten Märzwoche festlegen mußten. Noch sind Zimmer frei, so dass wir Teilnehmer nachmelden können. Alle Anmeldungen werden wir nach dem Prinzip "first come, first served" bearbeiten.

Heinz Schnitter