FORTH — die etwas andere Programmiersprache

Stellen Sie sich eine Programmiersprache vor, interaktiv wie Basic, schnell wie Assembler, strukturiert wie Pascal und beliebig erweiterbar. Eine solche Sprache ist Forth.

Forth wurde Anfang der siebziger Jahre von Charles H. Moore entwickelt und ursprünglich zur Steuerung von Radioteleskopen eingesetzt. Die Sprache entstand dabei ganz gezielt als Alternative zur fehleranfälligen und schlecht zu wartenden Assemblerprogrammierung auf der einen Seite und den klassischen Compilersprachen wie Fortran oder Algol auf der anderen Seite, die für Prozeßsteuerungen zu langsam und zu aufwendig waren.

Forth wurde übrigens damals als sogenannte »Programmiersprache der vierten Generation« entwickelt und sollte eigentlich dementsprechend den Namen »Fourth« tragen. Leider war die IBM 1130, auf der Forth zum ersten Mal implementiert wurde, noch ein Rechner der »dritten Generation« und erlaubte nur Dateinamen bis maximal fünf Zeichen. Charles Moore mußte »Fourth« daher um einen Buchstaben kürzen, und das war die Geburtsstunde von Forth. Lange Jahre fristete die Sprache ein Schattendasein in diversen Forschungsstätten, ehe sie mit dem Aufkommen der Mikrocomputer eine größere Verbreitung auch unter Privatleuten fand.

Was ist Forth?

Forth ist interaktiv. Ähnlich wie in Basic können Programme also im Dialog mit dem Computer entwickelt und getestet werden. Das ist ein wesentlicher Unterschied zu den klassischen Compilersprachen wie zum Beispiel Pascal oder Fortran. Ein Compiler erwartet nämlich ein fertiges Programm, das er dann in einem Arbeitsgang übersetzt. Es gibt keine Möglichkeit, die Wirkung bestimmter Kommandos vor der Übersetzung zu testen. Anders bei Forth. Hier kann man im Direktmodus Rechnungen durchführen, Programmteile testen und neue Befehlsworte definieren.

Forth ist strukturiert. Es gibt in Forth selbst mit Gewalt keine Möglichkeit, Basic-ähnlichen Spaghetti-Code zu erzeugen. Der Grund dafür ist sehr einfach: Es gibt keine GOTO-Befehle und auch keinen Ersatz dafür. Jede Wort-Definition ist in sich abgeschlossen und am ehesten noch mit den Prozeduren in Pascal zu vergleichen.

Forth ist schnell. Alle Anweisungen werden zuerst compiliert und dann ausgeführt, was insbesondere bei Programmschleifen große Zeitvorteile bringt. Forth-Programme sind daher in der Regel um Größenordnungen schneller als entsprechende Basic-Programme.

Forth ist erweiterbar. Im Gegensatz zu den meisten anderen Programmiersprachen kann der Benutzer in Forth neue Sprachelemente definieren, ja Programmierung in Forth besteht gerade in der Definition neuer Worte zur Erweiterung des Sprachumfangs.

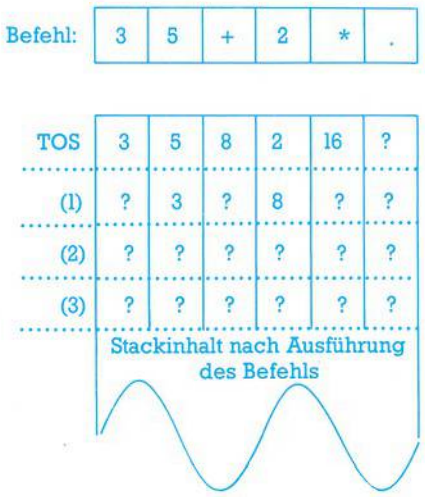

Bild 1. Die Stapelveränderungen während der Ausführung einer einfachen Befehlsfolge. Die Operationen »+« und »*« verknüpfen jeweils die beiden obersten Elemente des Stacks miteinander und legen das Ergebnis wieder ab. Mit ».« wird der TOS ausgedruckt und steht danach nicht mehr zur Verfügung.

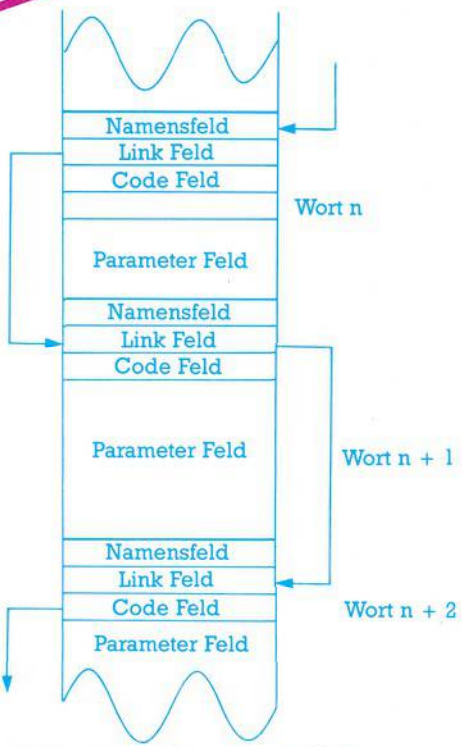

Bild 2. Vereinfachte Darstellung des Forth-Wörterbuchs im Speicher. Das Namensfeld enthält den Wortnamen, das Codefeld beinhaltet Information über die Art des folgenden Parameterfeldes, das die Definition des Wortes enthält. Ein Wort kann als Maschinenspracheabschnitt, als Variable oder Konstante oder — der häufigste Fall — durch andere Befehlsworte definiert sein (Colon-Definition). Das Link-Feld schließlich verkettet die Wortdefinitionen in Forth zu einer Liste, die bei der Interpretation oder Compilation von Worten durchsucht wird.

Jeder Basic-Programmierer wird erfreut zur Kenntnis nehmen, daß in Forth keine Fehlermeldung »SYNTAX ERROR« existiert — und zwar ganz einfach deswegen, weil die Forth-Syntax so einfach ist, daß keine solchen Fehler vorkommen können. Ein Wort in Forth besteht aus einer beliebigen Zeichenfolge, die allerdings kein Leerzeichen enthalten darf, weil dieses gerade zwei Worte trennt. Zum Beispiel sind gültige Forth-Worte

HALLO, 3FACH, %X!, +1NOCH-UND-DANN-SCHLUSS.

Eine Eingabezeile besteht nun einfach aus Worten und Zahlen. Bei Zahlen ist übrigens noch eine Besonderheit zu beachten: Forth kann

in beliebigen Zahlensystemen rechnen. Die Systemvariable BASE enthält die jeweils gültige Zahlenbasis. In der Regel wird man entweder im Dezimal- oder im Hexadezimalsystem arbeiten. Daher kennt Forth zwei spezielle Befehlsworte zum umschalten in das jeweilige Zahlensystem, nämlich DECIMAL und HEX.

Programmierung in Forth besteht nun einfach darin, mittels bereits definierter Worte wiederum neue Worte zu bilden. Ein Grundvokabular von vordefinierten Worten, das sogenannte Forth-Kernal, steht dem Benutzer dabei von Anfang an zur Verfügung.

Bevor wir näher auf die Programmierung eingehen, müssen wir uns zuerst noch mit einer weiteren Besonderheit von Forth auseinandersetzen, dem Stack. Forth arbeitet nämlich in umgekehrter polnischer Notation (UPN). Wesentliches Merkmal dieser Methode, die Benutzern von Hewlett-Packard-Taschenrechnern nicht unbekannt sein dürfte, ist ein Datenstack, auf dem alle Zahlenwerte gespeichert werden, bevor mit einem Operator auf sie zugegriffen wird. Um zum Beispiel den Wert des Ausdrucks 2*(3+5) zu berechnen, würde man in Basic einfach schreiben

```
PRINT 2*(3+5)
```

Dieselbe Berechnung in Forth lautet (Bild 1):

```
3 5 + 2 * . (man beachte den Punkt !)
```

Die Berechnung geht also genauso vor sich, wie man auch im Kopf rechnen würde: Zunächst nimmt man die beiden Zahlen in der Klammer, weil diese zuerst berechnet werden. Dann werden die Zahlen addiert, schließlich holt man sich die »2« und multipliziert sie mit dem ersten Zwischenergebnis. Forth legt also generell erst die für einen Befehl benötigten Operanden auf den Stack, und der Befehl greift anschließend darauf zu. Das gilt nicht nur für arithmetische Operationen, sondern ist ein Kennzeichen für die gesamte Sprachstruktur von Forth. Nehmen wir etwa den Punkt am Ende der Beispielrechnung oben. Er ist das Forth-Äquivalent zum PRINT-Befehl in Basic.

Beachten Sie bitte, daß der PRINT-Befehl immer vor den auszudruckenden Daten steht, der Punkt in Forth jedoch immer dahinter. Mit dem Punkt-Befehl wird stets der oberste Zahlenwert auf dem Stack ausgedruckt. Der wesentliche Unterschied zu anderen Programmiersprachen, wie zum Beispiel Basic, liegt nun darin, daß der Operand zum Zeitpunkt des Erreichens des Punkt-Befehls bereits bekannt ist. Der Basic-Interpreter, der auf einen PRINT-Befehl stößt, weiß zu diesem Zeitpunkt ja noch gar nicht, was er ausdrucken soll. Also muß die Information, daß gedruckt werden soll, irgendwo abgespeichert werden. Basic und alle anderen höheren Programmiersprachen bedienen sich dazu eines internen Stacks, der ähnlich wie der Datenstack von Forth konstruiert ist.

+	Addition
—	Subtraktion
*	Multiplikation
/	Division
MOD	Divisionsrest
/MOD	Rest und Ergebnis bei Division
MAX	Maximalwert
MIN	Minimalwert
ABS	Absolutwert
MINUS	Vorzeichenwechsel
AND	Logisches UND
OR	Logisches ODER
XOR	Logisches EXCLUSIV ODER
<	Test ob kleiner
>	Test ob größer
=	Test ob gleich
0<	Test ob negativ
0=	Test ob Null

Tabelle 1. Die wichtigsten arithmetischen und logischen Befehle in Forth

Letzten Endes liegt also der Unterschied zwischen Forth und anderen Sprachen darin, daß alle anderen Sprachen den vorhandenen Stack vor dem Benutzer verheimlichen und zu diesem Zweck natürlich zusätzlichen Verwaltungsaufwand treiben müssen, was die Verarbeitung nicht gerade beschleunigt.

Da Forth mit einem Datenstack arbeitet, funktionieren alle eingebauten und selbstdefinierten Funktionen nach dem gleichen Schema: Die Funktion holt sich die benötigten Parameter vom Stack, führt die notwendigen Berechnungen durch und legt das Ergebnis wieder auf den Stack. Dadurch können Funktionsaufrufe beliebig geschachtelt werden. Eine Übersicht über die wichtigsten numerischen Funktionen in Forth gibt Tabelle 1.

Da der Stack bereits für die laufenden Rechnungen benötigt wird, ist es einigermaßen umständlich, ihn auch noch für die Speicherung von Daten zu verwenden. Forth kennt daher die beiden Befehlsworte VARIABLE und CONSTANT, die zur Definition von Variablen beziehungsweise Konstanten benutzt werden. Mit 4 CONSTANT VIER wird zum Beispiel eine Konstante mit Namen VIER und Wert 4 angelegt, auf die im weiteren Verlauf immer wieder zurückgegriffen werden kann. Wann immer Forth jetzt auf das Wort VIER stößt, wird die Zahl 4 auf den Stack gelegt.

Bei Variablen ist die Situation etwas komplizierter. Sei eine Variable TEST mit Anfangswert 1 definiert worden durch 1 VARIABLE TEST. Wenn jetzt wieder das Wort TEST auftaucht, wird nicht der Inhalt der Variablen, sondern deren Adresse auf den Stack gelegt. Mit zwei sehr häufig gebrauchten Befehlen kann man nun auf diese Adresse zugreifen: Es sind dies die Befehle »!« (gesprochen »Store«) und »@« (gesprochen »Fetch«). Der Befehl »!« speichert den Zahlenwert, der an zweiter Stelle auf dem Stack liegt, an die Adresse, die durch das oberste Element des Stapels (TOS = Top of Stack) gegeben ist. Beide Zahlenwerte werden dabei vom Stack entfernt. Die Wirkung ist analog zum POKE-Befehl in Basic, operiert aber mit 16-Bit-Worten statt mit einzelnen Bytes. 2 840 ! entspricht also POKE 840,2:POKE 841,0 in Basic. Ebenso ist 840 @ in Forth analog zur Basic-Funktion PEEK(840)+256*PEEK (841). Sollen tatsächlich nur einzelne Bytes gespeichert und gelesen werden, bedient man sich der Worte »C!« und »@«. 1 840 C! entspricht POKE 840,1 in Basic und 840 C @ entspricht PEEK(840).

Nun sollte auch das Arbeiten mit Forth-Variablen klar sein: Bei jedem Auftreten eines Variablennamens legt Forth die Adresse dieser Variablen auf den Stack. Um zum Beispiel einer Variablen TEST den Wert 7 zuzuweisen, schreibt man in Forth

```
7 TEST !
```

Neue Worte definieren

Danach ist die Zahl 7 und auch die Adresse von TEST vom Stack verschwunden. Benötigt man die Zahl jedoch noch für weitere Rechnungen, muß sie vor der Zuweisung an die Variable zunächst dupliziert werden. Hierzu dient das Wort DUP, welches ein Duplikat des TOS erzeugt. DUP ist eine sehr häufig benutzte Funktion, da alle Forth-Befehle die Parameter vom Stack entfernen. Will man zum Beispiel das Zwischenergebnis einer längeren Rechnung mit dem Befehl ».« ausdrucken, dann muß es zunächst mit DUP dupliziert werden, da es durch ».« vom Stack gelöscht wird und somit für weitere Rechnungen nicht mehr zur Verfügung steht.

Forth hat die bemerkenswerte Eigenschaft, daß der Sprachumfang

beliebig erweitert werden kann. Zwei Möglichkeiten zur Schaffung neuer Worte haben wir ja schon kennengelernt: CONSTANT und VARIABLE. Daneben gibt es noch eine Reihe weiterer sogenannter Definitionsworte. Zum Beispiel können mit CREATE neue Forth-Worte direkt als Maschinenspracheroutinen definiert werden (sogenannte »Primitives«). Man wird davon allerdings nur an besonders zeitkritischen Stellen Gebrauch machen und in der Regel neue Worte mittels der sogenannten »Colon-Definition« einführen. Die Syntax dieser Definitionsmethode ist die folgende:

: NEUWORT ALTWORT1 ALTWORT2 .... ALTWORTn ;

Diese Befehlssequenz erzeugt ein neues Wort, NEUWORT, welches durch die bereits vorhandenen Worte ALTWORT1 bis ALTWORTn definiert ist. Die Colon-Definition beginnt also immer mit dem Wort »:« und endet mit dem Wort »;«. Es ist wichtig, sich genau klar zu machen, daß »:« und »;« nicht irgendwelche syntaktisch notwendigen Zeichen sind, sondern daß es sich dabei tatsächlich um normale Forth-Worte handelt. Das Wort »:« ruft vereinfacht gesagt den Forth-Compiler auf, während das Wort »;« eine Anweisung zum Beenden der Compilation darstellt.

Hinsichtlich der Namensgebung für neue Worte ist der Anwender im Prinzip keinen Beschränkungen unterlegen. Benötigt man zum Beispiel des öfteren den Durchschnitt zweier Zahlen, dann kann man einfach definieren

: MITTEL + 2 / . ;

Nach dieser Definition kann man jetzt einfach schreiben 4 8 MITTEL und erhält den Wert 6 angezeigt. MITTEL erwartet also zwei Zahlen auf dem Stack, die zunächst addiert werden (»+«). Das Ergebnis dieser Operation wird dann durch zwei geteilt (»2 /«) und schließlich ausgedruckt (».«). An der Definition von MITTEL ist schon zu sehen, wie kurz und effizient Forth-Programme sein können. In Basic würde ein entsprechendes Programm etwa folgendermaßen aussehen:

10 INPUT X,Y : PRINT (X+Y)/2 : REM Mittel

Mit der Colon-Definition kann man auch auf einfachem Wege Forth-Worten neue Namen zuweisen. Wenn man zum Beispiel für das Ausdrucken einer Zahl lieber den Befehl »DRUCKE« hätte, kann man einfach definieren

: DRUCKE . ;

Nun kann man überall DRUCKE statt ».« schreiben. Wenn man es wieder leid ist, ständig sechs Buchstaben statt einem schreiben zu müssen, teilt man dem Forth-System einfach mit, daß es das Wort DRUCKE vergessen soll. Der dazu nötige Befehl lautet FORGET DRUCKE. Allerdings löscht dieser Befehl nicht nur die Definition von DRUCKE, sondern auch alle anderen eventuell noch danach erfolgten Definitionen. Das hängt damit zusammen, daß Forth alle Definitionen in einer verketteten Liste abspeichert, dem sogenannten Wörterbuch (Bild 2).

Strukturiert Programmieren

Forth zwingt den Programmierer geradezu, seine Programme modular aufzubauen. Es gibt in Forth keinen dem GOTO in Basic vergleichbaren Befehl. Ein komplexes Forth-Programm entsteht immer zuerst auf dem Papier. Wenn die Programmstruktur festliegt, entwickelt man zunächst die benötigten Unterroutinen und testet sie aus. Diese Unterroutinen sind danach ja als völlig normale Forth-Worte zu benutzen und können dazu verwendet werden, nun eine weitere Ebene noch »mächtigerer« Worte zu definieren. Am Ende dieses Prozesses steht dann praktisch ein Wort, das das gesamte Programm repräsentiert.

Forth besitzt, ähnlich wie Pascal, eine ganze Reihe sogenannter Kontrollstrukturen, um strukturierte Programmierung zu unterstützen:

IF ... ENDIF — Die IF-Abfrage testet, ob der TOS (Top of Stack, also das oberste Element des Stacks) den Wert Null (= False) oder einen Wert ungleich Null (= True) hat. Ist das Ergebnis »True«, dann werden die Anweisungen zwischen IF und ENDIF ausgeführt, andernfalls wird das Programm nach ENDIF fortgesetzt. Einige ältere Forth-Versionen benutzen das Schlüsselwort THEN statt ENDIF.

IF ... ELSE ... ENDIF — Arbeitet ähnlich der einfachen IF-Abfrage, führt aber im Falle TOS=0 die Anweisungen zwischen ELSE und ENDIF aus, sonst die Anweisungen zwischen IF und ELSE.

DO ... LOOP — Entspricht der FOR-NEXT-Anweisung in Basic. Das Wort DO erwartet zwei Parameter auf dem Stack, nämlich die Anfangs- und die Endzahl der Schleifendurchläufe. Mit LOOP wird die Schleife beendet. Falls das Wort +LOOP statt LOOP verwendet wird, erhöht sich der Schleifenindex nicht um 1, sondern um den Wert des TOS. Das Wort »I« wird innerhalb von Schleifen dazu verwendet, den Schleifenzähler auf den Stack zu kopieren.

BEGIN ... UNTIL — Diese Schleife wird solange durchlaufen, bis UNTIL einen Wert ungleich Null (also »True«) auf dem Stack vorfindet. Entspricht REPEAT ... UNTIL in Pascal.

BEGIN ... WHILE ... REPEAT — Der Programmteil zwischen BEGIN und REPEAT wird solange durchlaufen, bis das Wort WHILE die Bedingung TOS=0 feststellt. Danach wird die Schleife unmittelbar hinter WHILE verlassen.

BEGIN ... AGAIN — Dient zum Programmieren unendlicher Schleifen und testet keinerlei Bedingungen.

Wer in seinen Forth-Programmen weitere Schleifen- oder Auswahlstrukturen benötigt, kann sie sich einfach selber definieren. Das gilt für praktisch alle Bereiche. Forth arbeitet zum Beispiel normalerweise nur mit Integer-Arithmetik. Wer nun für seine spezielle Anwendung Real-Zahlen benötigt, besorgt sich einfach ein REAL-Vokabular von Forth und bindet es in sein System ein. So kann sich jeder Anwender sein persönliches Forth-System zusammenstellen und nach seinen Bedürfnissen von Fall zu Fall erweitern.

Es ist klar, daß es sehr schwierig ist, für eine derartig flexible Sprache einen sinnvollen Standard zu finden. Die Forth Interest Group (FIG), eine nichtkommerzielle Vereinigung von Forth-Enthusiasten, bemüht sich seit Jahren mit gewissem Erfolg um eine Standardisierung von Forth. Zu diesem Zweck gibt die Forth Interest Group vollständige Assemblerlistings des sogenannten FIG-Forth mit Implementationshinweisen für alle gängigen Mikrocomputer heraus. Eine spezielle Forth-Zeitschrift, die »Forth Dimensions« kann dort ebenfalls bezogen werden. Die Adresse ist Forth Interest Group, P.O. Box 1105, San Carlos, CA 94070, USA.

Wer sich näher mit Forth beschäftigen will, dem seien zum Abschluß noch drei Bücher empfohlen:

Starting Forth von Leo Brodie, zu beziehen über die Forth Interest Group;

Das Forth Handbuch von H. Floegel, erschienen im Hofacker-Verlag;

Die Programmiersprache Forth von R. Zech, erschienen im Franzis-Verlag. (ev)