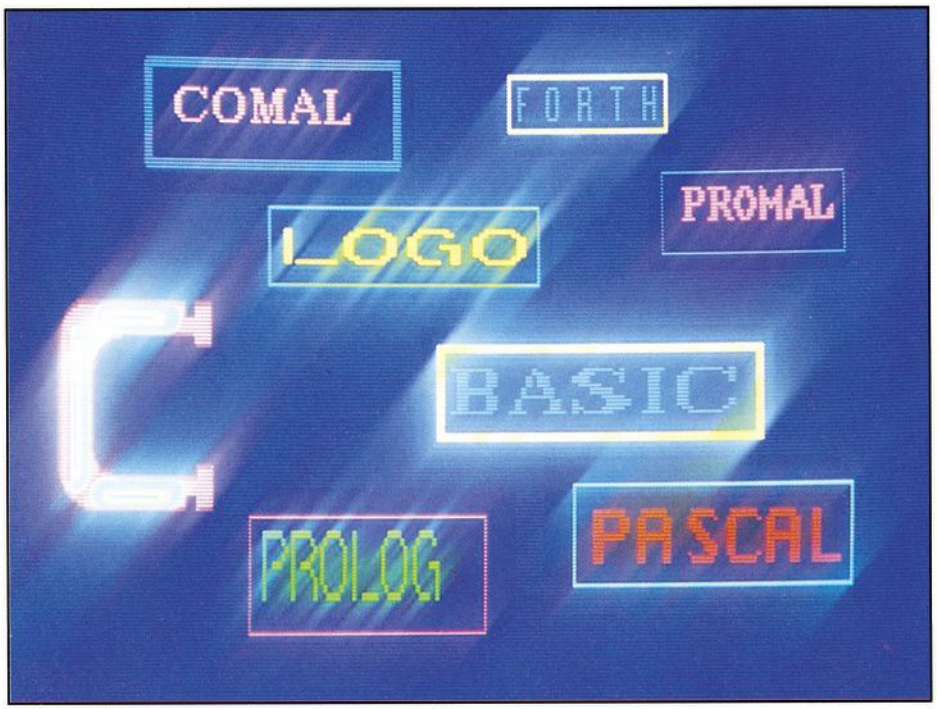
In der Sendereihe »Computerzeit« befaßt sich die nächste Sendung mit dem Thema »Programmiersprachen«. Im ersten Programm am 5. März 1986 um 16 Uhr 55 Minuten strahlt die ARD diesen Beitrag aus. Der folgende Artikel soll eine Ergänzung zu dieser Sendung darstellen.

Jede Verständigung, ob zwischen zwei Menschen oder zwischen einem Menschen und seinem Computer, erfolgt über Sprache. In der Kommunikation zwischen Menschen sind Deutsch, Englisch oder Chinesisch übliche Sprachen — je nachdem, aus welchem Land die Gesprächspartner stammen. In der Kommunikation von Computer und Mensch sind Sprachen wie Basic, Pascal, Cobol oder C bekannt und wichtig.

Die verschiedensten Programmiersprachen wurden im Laufe der Zeit entwickelt. Je nach Problemstellung und Computertyp benötigten die Programmierer bestimmte Befehle (zum Beispiel maschinennahe Befehle, Grafikbefehle, doppelgenaue Arithmetikbefehle für exakte, numerische Berechnungen), unterschiedliche Datentypen oder verschiedene Programmstrukturen (das Blockkonzept in Pascal, Verbundtypen in Ada). Die Forscher und Programmentwickler haben daraufhin die passenden Programmiersprachen entworfen. Denn ein Computer mit kleinem Speicher muß anders programmiert werden als ein Großrechner. Zum Commodore 64 wird deshalb standardmäßig der Basic-Interpreter mitgeliefert. Programmpakete auf größeren Computern wird man dagegen nicht in Basic, sondern eher in Pascal oder C entwickeln, weil der strukturierte Programmaufbau eines Pascal-Programms oder die Schnelligkeit eines C-Programms gefragt sind.

In diesem Artikel sollen die bekanntesten Programmiersprachen kurz vorgestellt und charakterisiert werden. Besonders ausführlich werden die höheren Programmiersprachen besprochen, die für den C 64 zu haben sind. Für die bekannten Programmiersprachen wie Lisp, Ada, Cobol oder Modula, die für den C 64 zuviel Speicherplatz benötigen, werden ihre typischen Eigenschaften und Vorzüge erklärt.




# Programmiersprachen

**Nur mit dem richtigen Werkzeug kann man optimal arbeiten. Was dem Handwerker die Arbeitsgeräte sind, das sind dem Programmierer Serviceprogramme — und Programmiersprachen.**

Auch im »Gespräch« zwischen Maschine und Mensch kommt es darauf an, woher die beiden Gesprächspartner stammen. Nicht alle Computer verstehen die verschiedenen Programmiersprachen gleich gut.

## Wie sag ich's dem Computer?

Ein Mensch arbeitet mit einem Computer, damit dieser ihm Arbeit abnimmt oder bestimmte Probleme löst. Solche Aufgaben teilt man dem Computer in Form von Programmen mit. Und von der Art des Problems hängt ab, wie der Anwender die Aufgabe formuliert. Niemand kann auf allen Gebieten gleich gut sein. Und genauso ist nicht jede Programmiersprache für alle Gebiete gleich gut geeignet. Es wurden Sprachen entwickelt, mit denen vor allem gerechnet werden sollte (Fortran, Basic). Andere sind für computerungeübte Kaufleute geeignet (Cobol). Pascal wiederum erzieht durch das Blockkonzept und den logischen und konsistenten Aufbau der Sprache dazu, Probleme (vor dem Programmieren) gründlich zu analysieren und einen strukturierten Denkansatz zur Problemlösung zu suchen. Da Pascal außerdem noch sehr leistungsfähig ist, wird diese Sprache oft als Lernsprache für Informatiker und zur Entwicklung umfangreicher Programmsysteme eingesetzt. Im Zug der Künstlichen Intelligenz-Forschung (KI) wurden schließlich spezielle Programmiersprachen entwickelt, um selbstlernende und sogar intelligente Programme zu schreiben. Die Standardsprachen in der KI sind Lisp und Prolog. Ada ist eine Sprache, die in letzter Zeit immer häufiger auftaucht. Sie wurde entworfen, um bei der Entwicklung sehr großer Programmsysteme Sicherheit und die Freiheit von Programmierfehlern zu ermöglichen. Zuerst sollen

nun die wichtigsten Programmiersprachen, die auf Großrechnern verfügbar sind, ganz kurz skizziert werden. Damit man sieht: auch außerhalb der C 64-Welt wird programmiert — und nicht schlecht.

## Programmiersprachen einer anderen Welt

Ada wurde entwickelt, um eine zuverlässige und »sichere« (einfache Überprüfbarkeit auf Bugs) Sprache zu schaffen, die vor allem im militärischen Bereich eingesetzt werden kann. Besondere Features von Ada sind Prozesse, die quasiparallel ablaufen und Module, das sind Programmbausteine, in denen logische Programmteile so zusammengefaßt werden können, daß Variablenwerte von außerhalb des Moduls nicht verändert werden können. Fehler durch versehentliches Überschreiben werden so ausgeschlossen. Der Ada-Compiler braucht sehr viel Platz. Daher ist eine vollständige Version dieser Sprache auf Heimcomputern nicht zu haben.

Algol 60 (algorithmic language) wurde Anfang der 60er Jahre für den technisch-wissenschaftlichen Bereich entwickelt. Als erste höhere Programmiersprache ließ sie strukturierte Programmierung zu. Algol verfügt über eine Blockstruktur. Sprünge, Laufanweisungen und Prozeduren stehen zur Verfügung. Algol wird heute kaum noch zum Programmieren eingesetzt. Algol spielt in der Geschichte der Programmiersprachen eine ganz wichtige Rolle, weil sie die Grundlagen einer ganzen Klasse von Programmiersprachen liefert: der blockstrukturierten Sprachen wie Pascal. Auch die Entwicklung von Ada wurde von diesem Prinzip entscheidend beeinflußt.

APL (A Programming Language) wurde an der Harvard-Universität als vereinfachte Beschreibungssprache für mathematischen Strukturen und Operationen entwicklelt. APL verfügt nicht über die klassischen Daten- und Programmstrukturen, sondern verwendet Felder als grundlegende Datenstruktur und spezielle Feld-Operationen zur Verarbeitung der Daten. Der Befehlsvorrat von APL besteht aus einer Vielzahl von mathematischen und logischen Operationen. Bedingte Anweisungen und Schleifen sowie Sprachelemente zu Listenverarbeitung fehlen dagegen. APL ist mehr von wissenschaftlichem Interesse. In der kommerziellen Programmierung ist diese Sprache nicht verbreitet.

Cobol (COmmon Business Oriented Language) wurde speziell für kaufmännische und wirtschaftliche Aufgaben entwickelt. Diese Programmiersprache wurde aus der englischen Umgangssprache entwickelt. Mit vielen Worten beschreibt ein Cobol-Programm, was getan werden soll. Die Programme sind selbstdokumentierend, daher sind die Programme relativ leicht lesbar.

Fortran wurde 1956 entwickelt und wird vor allem im technisch-wissenlichen Bereich noch immer sehr häufig verwendet. Statistische Auswertungen für Diplomarbeiten oder andere numerische Berechnungen werden vorwiegend in Fortran programmiert. Die bekanntesten Statistik-Programmpakete wurden in Fortran implementiert, ebenso eine große Anzahl von Software-Paketen. Basic kann als eine abgemagerte Version von Fortran angesehen werden, in der diejenigen Programmkonzepte gestrichen wurden, die platzaufwendig sind.

Lisp ist die bekannteste Sprache im Bereich der Künstlichen Intelligenz. Lisp ist eine listenorientierte, »funktionale« Programmiersprache, die seit etwa 1960 entwickelt und immer weiter modifiziert wurde. Vor einigen Jahren wurden spezielle Computer für Lisp gebaut, die sogenannten Lisp-Maschinen. Eine Lisp-Maschine ist ein Ein-Mann-Computer mit einem sehr großen Speicher, der nur Lisp versteht. Lisp-Programme laufen rekursiv ab und benötigen daher viel Platz. Rekursive Funktionen rufen sich selbst direkt oder indirekt auf (Funktion A ruft Funktion B auf, die wieder Funktion A aufruft). Durch die Verwendung von Rekursion können bestimmte Probleme leicht programmiert werden. Aber die Realisierung rekursiver Funktionen auf dem Computer ist sehr aufwendig und daher nicht in allen Programmiersprachen verfügbar.

Lisp-Maschinen werden in Forschungsinstituten der Universitäten und der Industrie zur Entwicklung von Expertensystemen eingesetzt. Da Programm und Daten dieselbe Struktur haben, können Lisp-Programme sich selbst verändern das heißt, sie können lernen! Daher hat Lisp in der Künstlichen Intelligenz eine führende Rolle bei der Programmentwicklung sogenannter »intelligenter« Programmsysteme eingenommen.

Ihr Computer kann mehr als Sie glauben, wenn Sie seine Fähigkeiten durch eine neue Programmiersprache erweitern. Sicher haben Sie sich schon über das dürftige Basic des C 64 geärgert. Vielleicht haben Sie sich deshalb schon einmal überlegt, auf eine andere Programmiersprache umzusteigen.

## Erweitern Sie die Fähigkeiten Ihres C 64!

Aber da beginnen die Probleme erst. Denn inzwischen gibt es auf dem Markt eine große Auswahl an Sprachen für den C 64. Ob nun eine Sprache auch das leistet, was man sich erhofft hat, merkt man aber erst, wenn man ein wenig damit programmiert hat. Hat man die falsche Sprache erwischt, ist der Frust groß und man kehrt zum guten alten Basic zurück.

Wir wollen Ihnen bei der Entscheidung weiterhelfen, welche Programmiersprache für Ihren Zweck die richtige ist, denn jede Sprache hat natürlich ihre Stärken und Schwächen. Im folgenden stellen wir Ihnen eine Auswahl der wichtigsten Programmiersprachen vor, die es für den C 64 gibt. In dieser Ausgabe finden Sie übrigens zum Thema Programmiersprachen auf dem C 64 einen Pascal-Kurs (Teil 1) für Basic-Programmierer, eine Beschreibung der Sprache C und einen Bericht über die Sprache Prolog auf dem C 64.

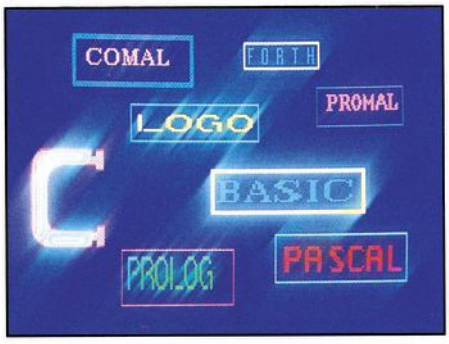
## Pascal

Die Sprache Pascal wurde von dem Schweizer Professor Nikolaus Wirth ins Leben gerufen. Sein Anliegen war es damals, besonders das strukturierte Programmieren und Denken zu fördern. »Spaghetti-Code«, wie es von Basic-Programmen her bekannt ist, gibt es in Pascal nicht. Eine strenge Strukturierung sorgt dafür, daß die Programme immer übersichtlich und gut lesbar sind. Aber nicht nur der Programmtext ist sauber gegliedert. Auch für die Variablen gibt es Strukturen. Bevor man sich an den Computer setzt, sollte das Programm bereits gründlich durchdacht sein: Welche Variablen brauche ich, wie kann ich diese gliedern, und nach welchen Grundgedanken soll das Programm strukturiert sein? Erst wenn dies alles klar ist, geht es ans Ausformulieren der einzelnen Routinen. Dieses Konzept hat durchaus seine Vorteile. Es treten weniger Fehler auf, da ja bereits eine Menge Überlegung in

das Programm eingeflossen ist. Das ist auch deshalb wichtig, weil Pascal eine Compilersprache ist, das heißt der Programmtext muß vor der Ausführung von einem Compiler erst einmal in ein Maschinenprogramm übersetzt werden. Bei vielen Fehlern kann durchaus das Austesten zu einer langwierigen Prozedur ausarten. Andererseits wird die Ausführung der Programme durch das Compilieren beschleunigt. Pascal-Programme sind deshalb in der Regel schneller als Basic-Programme.

**Wie sieht nun die Strukturierung in Pascal aus?** Ein Pascal-Programm besteht aus dem Hauptprogramm, das immer am Schluß des Textes definiert wird und beliebig vielen Prozeduren und Funktionen, die man am ehesten mit den Unterprogrammen in Basic vergleichen kann. Zeilennummern gibt es in Pascal nicht. Die Prozeduren und Funktionen werden mit ihrem Namen aufgerufen. Den ärgsten Feind jeder Strukturierung, den GOTO-Befehl, gibt es zwar in Pascal auch, er gilt aber als verpönt. Durch die Struktur-Anweisung REPEAT ... UNTIL, WHILE ... DO, CASE, FOR-Schleifen und IF ... THEN ... ELSE-Entscheidungen kann man sehr gut ohne GOTO auskommen.

Bei den Daten ist der Pascal-Programmierer gezwungen, sich genau zu überlegen, welche Variablen von welchem Typ er benötigt. Dies muß dem Compiler in Variablen- und Typendeklarationen mitgeteilt werden. Neben den von Basic her bekannten Typen Integer, Fließkomma und Zeichen gibt es in Pascal noch mehr Datentypen. Der Typ Boolean bezeichnet eine logische Variable, die nur die Werte für True und False annehmen kann. Der Typ SET ist für Mengen gedacht. In Mengen gibt es keine Reihenfolge der Elemente, wie zum Beispiel in einem Array, aber man kann zum Beispiel abfragen, ob ein bestimmter Wert in einer Menge enthalten ist. Daneben gibt es noch die strukturierten Datentypen Array und Record. Bei den Arrays handelt es sich um ein- oder mehrdimensionale Felder, wie wir sie von Basic her kennen. Ganz neu für den Basic-Programmierer dürfte aber der Typ Record sein. Damit können Variable verschiedenen Typs zu einer Verbund-Variablen zusammengefaßt werden. So können Daten sehr übersichtlich organisiert werden. Doch damit sind die Möglichkeiten von Pascal noch nicht ausgeschöpft. Der Typ Zeiger erlaubt ganz andere Dateistrukturen. Ein Zeiger ist eine Variable, die die




Adresse einer anderen Variablen enthält. Damit lassen sich verkettete Listen aufbauen, wobei jedes Element der Liste einen Zeiger auf das nächste Element der Liste enthält. Durch Ändern der Zeiger kann man beliebig Elemente einsortieren, anhängen oder wieder aus der Liste streichen. Eine ähnlich flexible Struktur ist die Baumstruktur, die auch mit Zeigern realisiert werden kann.

Wem diese Datentypen noch nicht reichen, der kann sich in Pascal noch eigene Typen definieren. Man kann beispielsweise den Typ Farbe deklarieren, der die Werte Rot, Grün oder Blau annehmen kann.

Sie sehen also, daß sich mit Pascal ganz neue Möglichkeiten auftun. Aber wie macht man aus dem C 64 eine Pascalmaschine? Es gibt inzwischen mehrere Pascal-Compiler, wir wollen uns hier aber auf die Versionen beschränken, bei denen nicht zu viele Abstriche vom Standard-Sprachumfang gemacht wurden.

**Pascal auf dem C 64**

Da kommen in Frage: Das KMMM Pascal, Oxford Pascal sowie Schtac Pascal, das in einer erweiterten Version auch von Data Becker als Profi Pascal vertrieben wird.

Oxford Pascal (Computer Plus Soft GmbH, Bahnstr. 22-26, 4220 Dinslaken, 199 Mark) unterstützt den vollen Sprachumfang und hat noch einige Extras zu bieten. So gibt es Grafik- und Soundbefehle, die von den Möglichkeiten des C 64 Gebrauch machen. Es ist sogar möglich, den Bildschirm in einen Grafikbereich und ein Textfenster zu unterteilen. Allerdings wird die Ausführung der Programme durch den dabei verwendeten Programmiertrick deutlich langsamer. Das Entwickeln von kleineren Programmen ist mit Oxford Pascal sehr angenehm. Editor und Compiler befinden sich im Speicher des Computers, so daß man ohne Diskettenoperationen gleich austesten kann. Erst bei längeren Programmen muß dann von Diskette compiliert werden.

Gegenüber Standard-Pascal wurde KMMM Pascal um einige Funktionen erweitert. Es gibt zum Beispiel einen Zufallsgenerator, POKE und PEEK, und erweiterte Möglichkeiten zur Stringverarbeitung, die vom Standard etwas stiefmütterlich behandelt wird.

Da das Nachladen mit der langsamen 1541-Floppy leicht zur Geduldsprobe werden kann, hat Data Becker bei seinem Profi Pascal (Data Becker, Merowingerstr. 30, 4000 Düsseldorf, 198 Mark) Routinen eingebaut, die das Nachladen um den Faktor drei beschleunigen. Nach dem Laden erscheint ein Menü, von dem aus der Editor, der Compiler und andere Funktionen angewählt werden können. Die notwendigen Programmteile werden dann nachgeladen. Profi Pascal enthält zusätzlich zum vollen Sprachumfang viele zusätzliche Funktionen. So ist der direkte Zugriff auf den Speicher des Computers möglich und der Typ String erlaubt bequeme Manipulationen von Zeichenketten. Um auch mit relativen Dateien effizient arbeiten zu können, was in Standard Pascal überhaupt nicht möglich ist, werden die Disketten mit einem eigenen Dateisystem organisiert. Dadurch können beliebige Datensätze mitten in einem File gelesen werden. Daneben bietet Profi Pascal die Möglichkeit, Assembler-Routinen direkt in das Pascal-Programm einzubauen.

## Forth

Ein völlig anderes Konzept als Pascal liegt der Sprache Forth zugrunde. In Forth dreht sich alles um das Stack-Prinzip. Der Stack ist ein Speicher, der nach dem »Last In First Out (LIFO)-Prinzip« arbeitet. Das heißt: der letzte Wert, der auf den Stack geschrieben wurde, kann als erster wieder vom Stack heruntergeholt werden. Sämtliche Rechenoperationen in Forth werden über den Stack abgewickelt. Wer schon einmal mit Taschenrechnern der Firma Hewlett-Packard gearbeitet hat, kennt das dabei verwendete Prinzip der umgekehrt polnischen Notation (UPN).

Eine weitere Eigenschaft von Forth ist es, daß der Sprachumfang beliebig erweitert werden kann. Aus bereits bestehenden Forth-Befehlen können neue Befehle kombiniert werden, die dann in Zukunft zur Verfügung stehen. Es ist sogar so, daß der größte Teil von Forth in Forth selbst geschrieben wurde. Nur ganz wenige elementare Befeh-

le sind in Assembler geschrieben, der Rest wurde aus diesen wenigen Worten aufgebaut. Durch dieses Baukasten-Prinzip kann sich jeder »sein« Forth selbst zusammenbauen.

**Wie arbeitet man nun mit Forth?** Forth arbeitet wahlweise mit Interpreter oder Compiler. Nach dem Start ist zunächst der Interpreter aktiv. Er bearbeitet ein Programm, ähnlich wie der Basic-Interpreter des C 64. Er holt sich immer das nächste Wort und versucht es auszuführen. Das kostet natürlich Zeit, und deshalb gibt es noch den Forth-Compiler. Durch einen Doppelpunkt erfährt das Forth-System, daß der folgende Text nicht interpretiert, sondern compiliert werden soll. Der Compiler macht daraus ein neues Forth-Befehlswort und trägt dieses in seine Liste ein. Von nun an steht das neue Wort dem Interpreter und dem Compiler zur Verfügung. Compilierte Worte machen Forth zu einer sehr schnellen Programmiersprache, die etwa zehnmal so schnell wie Basic ist.

Um Ordnung in den Programmablauf zu bringen, gibt es in Forth die Kontrollstrukturen IF..ELSE..ENDIF, DO..LOOP, BEGIN..UNTIL, BEGIN.. WHILE..REPEAT und BEGIN.. AGAIN. Ein GOTO gibt es in Forth überhaupt nicht.

Natürlich gibt es auch für Forth einen Standard, sozusagen eine Minimalausstattung für Forth-Systeme. Dieser Standard wurde von der Forth Interest Group geschaffen und heißt deshalb FIG-Forth. Die meisten Versionen für den C 64 enthalten allerdings weit mehr Befehle als der Standard, da sich Forth ja sehr leicht erweitern läßt. Wir wollen Ihnen einige Forth-Systeme für den C 64 vorstellen.

Das »64 Forth« erfüllt die Anforderungen des FIG Standards. Darüber hinaus bietet es eine Menge zusätzlicher, an den C 64 angepaßter Worte. Es stehen mehr als 500 Befehle zur Verfügung. Diese sind auf mehrere Vokabulare verteilt, die man einzeln aktivieren kann. Es gibt die Bereiche FORTH, EDITOR, ASSEMBLER und SYSTEM. Das FORTH-Vokabular enthält alle Worte, die man zum Programmieren braucht. Zum Eingeben größerer Programme dient das EDITOR-Vokabular. Mit SYSTEM stehen dem Anwender Befehle des Betriebssystemes zur Verfügung. Wenn es mal ganz schnell gehen soll, kann man mit ASSEMBLER Maschinenroutinen in die Forth-Programme einbauen.

**Grafik und Sound mit Forth**

Die Grafik- und Soundmöglichkeiten des C 64 werden von 64 Forth unterstützt. Sogar ein Sprite-Editor ist enthalten. Der Full Screen Editor ist eine angenehme Verbesserung des Standards, der nur zeilenweise Eingabe erlaubt.

Das »Super Forth 64« (Forth Systeme, Angelika Flesch, Schützenstr. 3, Titisee Neustadt, 398 Mark) enthält nicht nur den FIG-Standard, sondern insgesamt über 700 Worte. Je nach Bedarf kann man sich die Befehle zusammenstellen. Der Umgang mit Grafik und Musik wird durch die Befehle vereinfacht. Auch hier gibt es einen Sprite-Editor und als Krönung noch die sogenannte Turtlegrafik (siehe Logo). Das Rechnen mit Fließkommazahlen, das in Forth normalerweise nicht vorgesehen ist, wird durch ein eigenes Befehlspaket unterstützt. Die Steuerung von Interrupts, die man von Hochsprachen eigentlich gar nicht kennt, erlaubt Effekte wie einen geteilten Bildschirm oder parallel zum Programm laufende Soundeffekte. In einem Trace-Modus können Programme gründlich getestet werden.

Das »M & T-Forth« (Happy Software, Markt & Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar, 98 Mark) für den C 64 umfaßt nur etwa 280 Befehle, allerdings werden auf Diskette noch einige Forth-Programme mitgeliefert, die als Worte eingebaut werden können. Sprites, Grafik und Sound werden unterstützt, und der FIG-Standard ist voll enthalten. Damit ist die Kompatibilität zum Standard gegeben.

## Logo

Die Sprache Logo wurde vor allem durch die Turtle-Grafik (Commodore Händler, 159 Mark) bekannt. Dabei bewegt sich ein kleines Dreieck, die Turtle (zu deutsch Schildkröte) nach den Anweisungen des Programmierers über den Bildschirm und hinterläßt ihre Spuren in Form von Linien. Zur Steuerung gibt es die Befehle FORWARD, BACK, RIGHTTURN und LEFTTURN. Als Argumente werden die Länge der Strecke und bei den Turns ein Winkel angegeben. Mittels SETX und SETY oder SETXY kann man die Turtle auf definierte Ausgangspunkte setzen. Der Standort der Turtle kann durch XCOR und YCOR abgefragt werden. Durch diese Art der Grafiksteuerung lassen sich auf einfache Weise die tollsten Grafiken erzeugen. Aber Logo besteht nicht nur aus der Turtle-Grafik. Ein Logo-Programm ist aus mehreren Einzelprogrammen aufgebaut, also ein ähnliches Baukastenprinzip wie bei Forth. Es ist möglich, ein Programm in viele Teilaufgaben aufzuteilen und diese Bausteine dann zum eigentlichen Programm zusammenzufügen. Die Entwicklung eines Programms wird so überschaubar und strukturiert. Wie in Pascal kann sich übrigens ein Logo-Programm selbst aufrufen, und zwar mehrfach (Rekursion). Mit Rekursionen lassen sich viele Probleme äußerst elegant lösen. Ein anderes Element von Logo ist die Programmierung mit Listen. Eine Liste wird einfach durch eckige Klammern definiert. Durch verschiedene Listenbefehle können diese kombiniert, verglichen und bearbeitet werden. Listenelemente oder Teile können aus der Liste herausgenommen und zu neuen Listen oder Wörtern kombiniert werden. Mit den Listen können zum Beispiel Dateiverwaltungen recht einfach programmiert werden.

Die von Commodore selbst vertriebene Logo-Version für den C 64 bietet auch einige Kommandos für den Soundchip, um Tonhöhe, Tondauer und die Hüllkurve der Sounds zu bestimmen. Der Umgang mit Sprites wird ebenfalls erleichtert. Mit dem Sprite-Editor können Sprites entworfen werden, ohne sich, wie in Basic, mit Adressen und Hexadezimal-Zahlen herumschlagen zu müssen. Die Sprites können auf Diskette gespeichert und von dort wieder eingelesen werden.

Leider verbraucht Logo eine Unmenge an Speicherplatz im C 64. Das liegt daran, daß alle Befehle, auch die selbstdefinierten, im Speicher vorhanden sein müssen. Für das eigentliche Programm bleibt da oft wenig Platz.

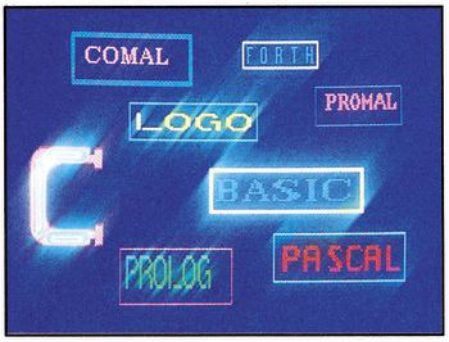
## Prolog

Der Name Prolog bedeutet PROgrammieren in LOGic. Damit ist schon gesagt, welches Konzept hinter der Programmiersprache Prolog steht: eine mathematische Methode der formalen Logik, die für automatisches Beweisen entwickelt wurde. Um zu beschreiben, wie Prolog arbeitet, muß man etwas weiter ausholen. Prolog-Programme »sagen« dem Computer nicht, was er tun soll. Sie beschreiben das, was der Computer »wissen« muß, um Probleme zu lösen. Diese Probleme werden vom Benutzer in Form von Fragen an den Computer gestellt. Prolog ist erst seit kurzem für den C 64 verfügbar (Brainware GmbH, Kirchgasse 24, 6200 Wiesbaden, 289 Mark). Diese

Version ist in dieser Ausgabe im Test. Prolog wurde dort ( auch für Basic-Anhänger) beschrieben.

## Comal

Wenn Sie bisher mit Basic einigermaßen gut zurechtgekommen sind, sollte Ihnen der Umstieg auf Comal eigentlich leichtfallen. Die Sprache ist stark an Basic angelehnt. Das heißt aber nicht, daß die Schwächen von Basic mit übernommen wurden. Auch bei Pascal wurden einige Anleihen gemacht, vor allem, was die Strukturierung betrifft. Von Logo wurde die Turtle-Grafik entliehen. Comal ist sozusagen eine Mischung der besten Elemente aus verschiedenen anderen Sprachen. Herausgekommen ist dabei eine leistungsfähige Sprache, die noch einen entscheidenden Vorteil hat: Einige Comal-Versionen (V.0.14) werden nämlich umsonst abgegeben (Comal 0.14, Interpool, c/o Prof. Leuschner, 7487 Gammetingen-Bronnen, 20 Mark, Comal 2.0, D. Belz, 2270 Utersum, 198 Mark), und es wird sogar dazu ermutigt, Comal zu kopieren und weiterzugeben!




Comal läßt sich weder als Compiler- noch als Interpretersprache bezeichnen. Die Wahrheit liegt irgendwo in der Mitte. Im Direktmodus kann man mit dem Interpreter arbeiten. Das Erstellen eines Programms läuft dagegen in drei Phasen ab. Die erste ist die Eingabe des Programms. Dabei tritt der sogenannte Syntax-Checker in Aktion. Er überprüft die eingegebenen Zeilen gleich auf syntaktische Fehler und gibt gegebenenfalls Fehlermeldungen aus. Das kann eine Menge an Fehlersuche ersparen. Im zweiten Durchgang wird das Programm nach Variablen und angesprungenen Zeilen durchsucht. Die Ergebnisse werden in einer Liste eingetragen, in der das Comal-System dann beim Programmlauf, der dritten Phase, nachschlagen kann. Das geht natürlich schneller als in Basic, wo der Interpreter den ganzen Programmtext durchsuchen muß, wenn er eine angesprungene Zeile sucht. Auch auf Variablen hat Comal durch die Liste eine schnelleren Zugriff als der Basic-Interpreter. Die Anweisungen werden aber in Comal nicht compiliert, sondern nach wie vor interpretiert. Von einer echten Compilersprache kann also nicht gesprochen werden.

Wie schon erwähnt, ist die Syntax von Comal stark an Basic orientiert. Aber dennoch wird ein Comal-Programm anders aufgebaut sein als ein Basic-Programm. Die an Pascal erinnernden Kontroll-Strukturen wie CASE, REPEAT.UNTIL und WHILE werden durch die (auch dem Basic-Programmierer geläufigen) Strukturen IF..THEN..ELSE und FOR..NEXT-Schleifen ergänzt. Mit LOOP..EXIT..ENDLOOP können auch Endlosschleifen konstruiert werden. Der GOTO-Befehl existiert in Comal zwar auch, sollte aber nur in Ausnahmefällen angewendet werden. Inzwischen gibt es mehre-

re Comalversionen für den C 64. Da sind zum einen alle Versionen, deren Versionsnummern mit einem Nuller beginnen, zum Beispiel Comal V.0.14. Diese Versionen können gegen einen geringen Unkostenbeitrag bezogen werden und beliebig weiterkopiert werden. Daneben gibt es noch kommerzielle Comal-Systeme, wie Comal 2.0. Diese enthalten einen größeren Befehlssatz als die Public Domain-Versionen.

## Promal

Promal ist eine stark strukturierte Sprache wie Pascal. Dennoch kann es von der Geschwindigkeit mit Forth mithalten und ist auch ähnlich maschinennah. Die Strukturierung erfolgt dabei durch Einrückungen im Programmtext, die das Programm gleichzeitig übersichtlich machen. Die üblichen Kontrollstrukturen IF..ELSE, FOR (ohne Next), CHOOSE (entspricht in etwa CASE in Pascal, REPEAT..UNTIL und WHILE) gibt es in Promal natürlich auch. Für zeitkritische Anwendungen kann man Maschinenroutinen aufrufen und dabei gleich Parameter übergeben, unter anderem auch über die drei Register des 6510-Prozessors. In den meisten Fällen wird man aber ohne Assembler auskommen, da Promal schon von Haus aus sehr schnell ist. Dies liegt unter anderem auch an den sehr schnellen Arithmetikroutinen, die die Routinen des Basic-Interpreters bei weitem übertreffen und dabei noch eine größere Genauigkeit haben. Aus den vier Grundrechenarten, die Promal beherrscht, kann man komplexere Berechnungen wie Exponential- oder Winkelfunktionen selbst programmieren und hat die Ergebnisse noch schneller als in Basic!

Das Promal-System besteht aus drei Teilen: dem Executer, dem Editor und dem Compiler. Der Executer ist ein komfortabler Kommando-Interpreter, von dem aus auch der Editor und der Compiler gestartet werden. Den Editor könnte man schon fast als Textverarbeitung bezeichnen. Er ist selbst in Promal geschrieben — ein weiterer Hinweis auf die Leistungsfähigkeit von Promal (Systems Management Associates, 3700 Computer Drive, Dept. GP, Raleigh, North Carolina 27609). Das Promal PM-200 kostet etwa 150 Mark ($ 49,95), die Entwickler-Version mit zusätzlichen Run-time-Programmen kostet etwa 300 Mark ($ 99,95). Für etwa 37 Mark ($ 12,50) gibt es die Demo-Version PM-200.

## C

In letzter Zeit gewinnt die Sprache C immer mehr an Bedeutung. Besonders in Verbindung mit dem Betriebssystem Unix hat C an Bedeutung gewonnen. Für den C 64 gibt es einen C-Compiler von Data Becker, der für 298 Mark fast den gesamten Sprachumfang bietet. So kann man auf dem »kleinen« C 64 mit C arbeiten. Markt & Technik bietet für 148 Mark (brandneu) das Smal-C-Entwicklungssystem mit Quellcode für den C 128. In diesem Heft finden Sie eine Einführung in die Sprache C. Auch auf den Compiler auf dem C 64 wird dort näher eingegangen.

Der Vorteil von C liegt hauptsächlich daran, daß man mit C maschinennah und damit schnell programmieren kann. Viele Anweisungen beziehen sich auf Programmiermethoden, die in Assembler häufig angewandt werden, wie das Rechnen mit Zeigern, Inkrementieren und Dekrementieren und das Arbeiten mit Bitfeldern. Letztere werden aber auf dem C 64 nicht unterstützt. Auch die Deklaration REGISTER, die eine Variable direkt in einem Register des Prozessors plaziert, gibt beim 6510-Prozessor mit seinen drei Registern keinen Sinn. Dennoch lassen sich in C auch auf dem Commodore 64 effiziente Programme schreiben. Ein weiterer Unterschied zu Pascal sind die Makros, die am Anfang eines Programms angegeben werden können. Darunter versteht man Befehlsfolgen, die durch Angabe ihres Namens im Programmtext eingesetzt werden können. Nun werden Sie vielleicht fragen, worin der Unterschied zu Unterprogrammen besteht, die ja auch nur eimal definiert werden und dann immer benutzt werden können. Ein Makro hat den Vorteil, daß der Compiler den Code für das Makro direkt ins Programm einsetzt. Sprünge und Parameterübergabe entfallen, was sich in der Geschwindigkeit auswirkt. Etwas ähnliches kennt man sonst nur von Assemblern.

### C hat eine eigene Philosophie

Wenn man sich den Befehlsvorrat von C ansieht, wird man erst einmal enttäuscht sein. Es gibt nur 13 Befehle. Nicht einmal ein Print-Befehl ist vorhanden. Aber es gehört zur offenen Philosophie von C, daß die benötigten Routinen aus externen Bibliotheken zum Programm dazugebunden werden. Erst dann entsteht ein lauffähiges Programm. Auf diese Weise kann C beliebig erweitert werden und der Speicher wird nicht mit unnötigen Befehlen belastet.

Beim Erstellen eines C-Programms geht man folgendermaßen vor: Zuerst erstellt man den Sourcetext mit dem Editor. Dieser Editor ist in der C 64-Version ein angenehmer Full-Screen-Editor, bei dem man sich sogar mit Farben Übersicht verschaffen kann. Dann betritt der Compiler die Szene und übersetzt das Programm. Um ein lauffähiges Programm zu erhalten, muß man danach den Linker auf das Compilat loslassen. Dieser bindet die Bibliotheksfunktionen dazu und stellt alle Bezüge her, die der Compiler noch offen gelassen hat. Erst jetzt kann das Programm getestet werden. Da die einzelnen Teile des C-Systems immer erst geladen werden müssen, kann das Übersetzen eines C-Programms die Geduld des Anwenders ganz schön auf die Probe stellen. Dennoch hat C viele Freunde, weil C-Programme sehr effizient sind.

## Was für wen?

Jetzt ist es an Ihnen zu entscheiden, welche Sprache für Sie die richtige ist! Wie Sie gesehen haben, gibt es fast für jeden Zweck eine geeignete Programmiersprache.

(Pehlandt/cg)

| | |
|---|---|
| Oxford Pascal | Computer Plus Soft GmbH, Bahnstr. 22-26, 4220 Dinslaken, 199 Mark |
| Profi Pascal | Data Becker, Merowingerstr. 30, 4000 Düsseldorf, 198 Mark |
| Schtac Pascal 64 | phs EDV-Beratung, Devenstedter Straße 8, 3000 Hannover 91, 798 Mark |
| 64 Forth | Forth Systeme Angelika Flesch, Schützenstr. 3, Titisee Neustadt |
| Super Forth 64 | Forth Systeme Flesch |
| M&T Forth | C 64-Software, Markt & Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar, 98 Mark |
| Logo | Commodore Händler, 159 Mark |
| Comal 0.14 | Interpool, c/o Prof. Leuschner, 7487 Gammetingen-Bronnen, 20 Mark |
| Comal 2.0 | D. Belz, 2270 Utersum, 198 Mark |
| Promal | Systems Management Associates, P.O. Box 20023, Raleigh, NC 27619, USA |
| C | Data Becker 298 Mark; Markt & Technik (C 128), 148 Mark |

**Tabelle 1. Bezugsquellen der Programmiersprachen**