

Facharbeit Informatik

Kurzer Abriß der
Geschichte
einiger wichtiger Programmiersprachen

Geschrieben von Michael Piefel
23.2.1991 – 3.3.1991

Einleitung

Im Informatikunterricht unserer Schule beschäftigen wir uns hauptsächlich mit der Programmierung. Unser Schwerpunkt liegt dabei bei zwei der weitverbreitetsten höheren Programmiersprachen, bei BASIC und Pascal. Leider kommt dabei ein Aspekt der Informatik zu kurz.

Die Informatik ist eine relativ junge Wissenschaft. Sie hat sich aus diesem Grund noch nicht richtig profilieren und auf eine andere Wissenschaft Einfluß nehmen können: auf die Geschichte. Anders als in der Mathematik, der Physik und der Chemie, deren größte Köpfe in die Geschichte eingingen – man denke nur an Newton oder Einstein, deren Namen auch derjenige schon gehört haben wird, der die Wörter Gravitation und Relativität nicht einmal schreiben kann –, sind die Großen der Informatik unbekannt, wer kennt schon Kernighan und Ritchie. Mit der Geschichte der Informatik muß sich die Informatik selbst beschäftigen, sie wird damit zu einer sehr umfangreichen Wissenschaft. Aber gerade das vergißt man leicht.

Der oben angesprochene Aspekt ist eben jener Zweig. Im Unterricht wird nur ein viel zu schmales Spektrum behandelt. Es ist natürlich, sich auf ein bis zwei Sprachen enger zu beziehen, noch dazu auf die Lehrsprache Pascal. Aber es müßten auch andere Sprachen, die eine große Bedeutung in bestimmten Gebieten haben wie z. B. COBOL, und Programmiertechniken wie OOP in den Unterricht Eingang finden und die Entwicklung der Informatik beachtet werden.

Denn die Informatik trennt etwas von fast allen anderen Wissenschaften: Sie wird vom Menschen bestimmt. Die meisten herkömmlichen bestehen zum größten Teil aus der Erforschung von in der Natur geschehenden Vorgängen und deren Interpretation. Die Informatik ist jedoch von Anfang an von Menschen erdacht und beeinflußt worden. Zum Verständnis dieser Wissenschaft ist es daher notwendig, sich ihre Geschichte vor Augen zu halten.

Eine Programmiersprache ist „... für die Kommunikation zwischen Mensch und Rechenanlage gedacht. Programme müssen jedoch erdacht und entwickelt werden. Programmierer unterhalten sich über sie ... In vielen Fällen werden Programme nie in maschinenlesbare Form gebracht, sie sind wichtig ... als Ansätze, ein neues Problem verstehen zu lernen.

Aus diesen Gründen ist abzuleiten, daß jede Programmiersprache eine soziale Erscheinung ist. ... Es ist stets eine gute Verfahrensweise gewesen, zum Studium einer sozialen Erscheinung ihre Geschichte zu studieren.“ /1/

Ich möchte mit meiner Arbeit einen kleinen Beitrag zur Verbreiterung des im Informatikunterricht vermittelten Wissens leisten.

Inhaltsverzeichnis

1	Prähistorie	5
2	Maschinensprache und Assembler	5
3	FORTRAN	5
3.1	Forderung	5
3.2	Erfüllung	5
4	BASIC	6
4.1	Name	6
4.2	Erfolglosigkeit	6
4.3	Wechselvolle Entwicklung	6
4.4	Wiederbelebung	6
5	Pascal und Modula	7
5.1	Pascal	7
5.2	Modula	8
6	C	8
6.1	Beliebtheit	8
6.2	UNIX	8
6.3	Aussichten von C	9
7	LISP	9
7.1	Namenserklärung	9
7.2	Problem und erste Lösung	10
7.3	Zufällige Entstehung	10
7.4	Metasysteme und LISP2	10
7.5	Zukunftschancen	10
8	Kurzüberblick über andere Sprachen	11
8.1	PROLOG	11
8.2	COBOL	11
8.3	PL/1	11
8.4	ALGOL	11

8.5 FORTH	11
8.6 Ada	12
9 Fazit	12
Sach- und Worterklärungen	13
Literaturverzeichnis	14
Beispielprogramme	15

1 Prähistorie

Man spricht von programmierbaren und nicht programmierbaren Rechnern. Aber diese Einteilung führt in die Irre. Denn eine Maschine ohne Programm leistet nichts. Das jedoch kann man von der Rechenmaschine Pascals nicht behaupten. So sollte man lieber in programmierbare und nicht mehr programmierbare Rechner einteilen. Das trifft eher den Kern der Sache. Die Programmierung der Vorläufer der heutigen Rechner hat nicht viel mit dem zu tun, was man heute unter Programmieren versteht. Eher kann man es mit dem Entwerfen eines Schaltkreises vergleichen. Trotzdem kann man es als Vorläufer heutiger Programmier-techniken und somit auch Sprachen ansehen.

2 Maschinensprache und Assembler

Der Prozessor eines jeden Computers versteht nur Befehle, die Zahlen im Binärsystem darstellen, also z. B. 110001010110. Diese Schreibweise ist fehlerträchtig, und man kann niemandem zumuten, sie zu benutzen. Darum wurde Assembler entwickelt. Das Wort steht hierbei sowohl für die Sprache als auch für das Programm selbst. Jedem Befehl in Maschinensprache ist dabei genau ein Befehl in Assembler zugeordnet. Man könnte Assembler auch als andere Schreibweise für Maschinensprache verstehen, was dazu geführt hat, beide Begriffe gleichzusetzen. Ein Befehl besteht dabei aus einer zwei- bis fünfstelligen Buchstabenkombination, die leicht – oder zumindest leichter als 110001... – einzuprägen ist und deshalb Mnemonik heißt. Assembler ist somit die älteste Programmiersprache überhaupt und entstand zeitgleich mit dem ersten programmierbaren Rechner. Als LLL ist sie abhängig vom Prozessor.

3 FORTRAN

3.1 Forderung

FORTRAN steht für FORMula TRANslation, also Formelübersetzung. Als die Forderung entstand, das mit dem Rechner zu tun, was sein Name vermuten läßt, das er kann, ohne daß für eine simple Wurzel 100 Zeilen Code verfaßt werden mußten, entstand auch die erste HLL: FORTRAN. Gleichzeitig war sie die erste problemorientierte Programmiersprache. Sie war zum Lösen von mathematischen Sachverhalten erschaffen worden. Sie ist der Großvater – oder besser die Großmutter – vieler folgender Programmiersprachen. Ganz neue Prinzipien wurden erst mit PROLOG, FORTH oder Smalltalk eingeführt.

3.2 Erfüllung

Im Jahre 1954 erschien der erste Sprachbericht in den USA von J. W. Backus, zwei Jahre später, 1956, war der erste Compiler verfügbar. Die Sprache hatte sofort riesigen Erfolg, war sie doch konkurrenzlos. Mitte der siebziger Jahre

erlebte sie jedoch einen Einbruch, Pascal lief ihr den Rang ab. Mit FORTRAN 77 wurde ein Standard geschaffen, der der Sprache half, wieder an Bedeutung zu gewinnen. Auch heute wird FORTRAN noch gerne zur Lösung von Problemen mathematischer Natur benutzt.

```
do 4k=i+1,n
  read (6,30,rec=k) name2,note2,du2
  if(du1.gt.du2) then
    write (6,30,rec=k) name1, note1, du1
  end if
continue
```

Programm 1: FORTRAN

4 BASIC

4.1 Name

Wie so viele Sprachen stammt BASIC direkt von FORTRAN ab. BASIC ist eine Abkürzung für Beginner's All purpose Symbolic Instruction Code, also etwa „Für alles geeigneter symbolischer Instruktionscode für den Anfänger“. Der Name ist heute diskriminierend. Gegen „symbolische Instruktionscode“ läßt sich natürlich nichts sagen, und auch „für alles geeignet“ ist entgegen aller Aussagen der Feinde von BASIC wahr. Die Diskriminierung besteht in dem Wort „Anfänger“ im Namen. Als die Sprache 1963 am Dartmouth College in den USA von John Kemeny und Thomas Kurtz entwickelt wurde, war das auch ihr geplanter Zweck.

4.2 Erfolglosigkeit

Sie sollte als Lehrsprache dienen, wogegen zwei Gründe sprachen: Sie war unstrukturiert und außerdem als Interpreter gedacht. Ein Interpreter ist zwar gerade zu Lernzwecken gut geeignet, aber er bedeutet, daß immer nur ein Teilnehmer den Computer für längere Zeit besetzte. Das Multitasking steckte damals noch in den Kinderschuhen.

4.3 Wechselvolle Entwicklung

Einen großen Auftrieb erhielt BASIC mit der Entwicklung von Mikrocomputern. In fast jedem war BASIC bereits im ROM eingebaut. Selbst zu MS-DOS, dem meistverbreiteten Betriebssystem, wird ein BASIC-Dialekt mitgeliefert. Aufgrund der fehlenden Strukturierung und den vielen verschiedenen Dialekten geriet die Sprache immer mehr in Verruf und drohte fast unterzugehen.

4.4 Wiederbelebung

Doch glücklicherweise bildete sich ein Standard heraus, so daß man heute Programme von einem Computer auf den anderen übernehmen kann, ohne viel zu ändern. Außerdem wurde die Sprache strukturiert. Heute gleicht die Sprache Pascal und C, ist jedoch einfacher zu erlernen. Die Ähnlichkeit wird noch durch die Tatsache verstärkt, daß BASIC heute fast nur noch als Compiler ausgeliefert wird. Mit den leistungsfähigen Dialekten – stellvertretend seien hier GFA-BASIC und MS-BASIC PDS genannt – ist es durchaus möglich, professionelle Programme zu schreiben, und einige Firmen tun dies auch.

5 Pascal und Modula

5.1 Pascal

5.1.1 Zielstellung

Auch Pascal stammt direkt von FORTRAN ab. Wie auch einige andere Sprachen wurde sie zu Lehrzwecken entwickelt und war ursprünglich nicht zum Schreiben größerer Programme gedacht. Man stand damals vor der Situation, daß es viele verschiedene Sprachen gab, meist für jedes Rechnersystem eine andere. Da man während der Ausbildung natürlich nicht wissen konnte, an welchen Maschinen die Schüler später arbeiten würden, entschied man sich an der Eidgenössischen Technischen Hochschule in Zürich, eine Lehrsprache zu entwickeln, anhand derer man den Schülern die Grundlagen des Programmierens nahelegen konnte.

5.1.2 Verwirklichung

Die gesamte Ehre an Pascal gebührt Nikolaus Wirth. Er entwickelte die Sprache in der Zeit von 1968 bis 1971. Wie die Schreibweise des Namens vermuten läßt, die nicht auf die übliche Namensgebung mit Hilfe von Abkürzungen hinweist, widmete Wirth seine Sprache dem genialen Mathematiker Blaise Pascal (1623-1662). Bereits 1970 wurde Pascal das erste Mal implementiert. 1971 stellte Wirth die Sprache der Öffentlichkeit vor.

Eine Sprache konnte zum damaligen Zeitpunkt – und eigentlich heute auch noch – nur überleben, wenn sie großzügige Unterstützung erhielt. Als bestes

Beispiel ist LISP zu nennen, die direkt bei IBM entwickelt wurde. Um so erstaunlicher ist der große Erfolg von Pascal. In Belfast wurde 1971 ein zweiter Compiler entwickelt. Nachdem Wirth 1972 einen revidierten Bericht der Sprache veröffentlichte – der übrigens nur Einschränkungen enthielt – und der dafür geeignete Compiler seit 1974 verteilt wurde, war der Erfolg von Pascal nicht mehr aufzuhalten. Pascal verdrängte zwischenzeitlich FORTRAN fast vom Markt.

```
readln(Buchstabe);  
for i:=1 to 15 do  
  if Buchstabe=Wort[i]  
  then begin  
    write(Buchstabe);  
    inc(Punkte)  
  end  
else write(' ');
```

5.1.3 Entzweigung der Standards und Turbo Pascal

Die große Verbreitung hatte aber auch den Nachteil, daß immer mehr verschiedene Pascal-Dialekte auftraten. Sie hatte damit die gleichen Probleme wie auch schon BASIC. Doch auch Pascal konnte „gerettet“ werden. Seit 1983 ist nämlich ein von Anders Hejlsberg programmierter Compiler auf dem Markt, Turbo Pascal. Er beseitigte den großen Nachteil der damaligen Compiler: Zuerst mußte man den Quellcode erfassen (erstes Programm), dann compilieren (zweites Programm), hatte der Compiler keine Fehlermeldung gegeben, so konnte man jetzt linken (drittes Programm). Turbo Pascal verband alle drei Komponenten und vereinfachte die Programmerstellung erheblich. Der große Erfolg des Compilers führte dazu, daß Turbo

Programm 2: Pascal

Pascal ein Standard wurde. Fast alle heute auf dem Markt existierenden Compiler (z. B. MS-Quick Pascal) orientieren sich an Turbo Pascal.

Turbo Pascal enthält seit 1989 auch OOP, womit das Überleben von Pascal zumindest für die nächsten Jahre gesichert ist.

Pascal gilt heute als Musterbeispiel der strukturierten Programmierung schlechthin, obwohl mit Modula eine besser strukturierte Sprache vorhanden ist.

5.2 Modula

Doch Wirth entwickelte seine Sprache weiter. Er führte eine Zahl neuer Konzepte ein und verbesserte Fehler, die er bei der Entwicklung von Pascal gemacht hatte. Aufgrund ihres modularen Aufbaus, der etwa mit dem Unit-Konzept Turbo Pascals vergleichbar ist, nannte er sie Modula. Größere Verbreitung erreichte erst Modula-2. Obwohl besser als Pascal und inzwischen auch mit extrem leistungsfähigen Compilern versehen (z. B. Top-Speed Modula-2) hat sie keinen so großen Erfolg. Wenn Borland oder Microsoft sich entschlossen, einen Modula-Compiler zu veröffentlichen, würde Modula gewiß einen Siegeszug antreten können.

6 C

6.1 Beliebtheit

C hat es geschafft, die Computerfreaks in zwei Lager zu spalten: Pro C und kontra C. Die einen beschwerten sich: „In Pascal nimmt Papa Wirth den Programmierer an die Hand und führt ihn über die Straße. In Basic darf er alleine laufen und in C sogar dann, wenn die Ampel rot zeigt.“, die anderen sagen, C sei die Abkürzung von Chaos. Beide Parteien haben recht. Die Sprache ist sehr knapp, vieles wird mit Sonderzeichen ausgedrückt. Diese Eigenschaft, die das Programmieren erleichtern sollte, macht Programme teilweise unübersichtlich. Ihre Möglichkeiten sind jedoch nahezu unbegrenzt und lassen dem Programmierer sehr große Freiheiten.

6.2 UNIX

6.2.1 Ziel

Die Entwicklung Cs ist eng mit der von UNIX verbunden. UNIX ist ein Betriebssystem, das 1969 zu entwickeln begonnen worden war. Es sollte ein System geschaffen werden, das auf allen Rechnern eine einheitliche Benutzerschnittstelle bot und außerdem leicht zu portieren sein mußte. Da es damals meist Rechenanlagen mit zwar guter Leistung, aber extrem hohen Kosten gab, so daß es kaum möglich war, den Rechner nur einer einzigen Person zur Verfügung zu stellen, sollte es sich um ein Multi-User-Multi-Task-System handeln.

6.2.2 Entwicklung von B und C

Die gute Portierbarkeit konnte nur dadurch erreicht werden, daß man eine HLL zur Programmierung benutzte. Diese jedoch waren für die Systemprogrammierung nicht geeignet. Man entwickelte also parallel zum Betriebssystem eine Sprache, die die Vorteile von LLL und HLL vereinigte, und schuf die erste MLL.

1969 begann man erste Assemblerprogramme für das neue Betriebssystem UNIX V1 zu schreiben. 1970 ging man dann mit UNIX V2 zur Sprache B über, welche ein Zwischenprodukt darstellte, und einigte sich auf den Namen UNIX. Mit UNIX V5 erreichte man schließlich die Sprache C. UNIX V7 war übrigens der erste Standard.

UNIX mußte anfangs noch in Assembler programmiert werden, mit den Sprachen B und C wurde das Betriebssystem dann neu implementiert. Durch diese Sprache (gemeint ist C, denn B war als Übergangsprodukt natürlich längst wieder verschwunden), deren spätere Compiler in ihr selbst geschrieben wurden, gelang es, das Betriebssystem sehr leicht von einem Rechner zum anderen übertragbar zu gestalten. Etwa 95-98% (die Angaben in der Literatur schwanken) von UNIX sind in C geschrieben, nur die restlichen 5% in Assembler, um die rechnerabhängigen Teile zu verwirklichen.

B wurde 1970 von K. Thompson konzipiert und nahm starken Einfluß aus BCPL, welche von M. Richards entwickelt wurde. C, die B.W. Kernighan und D.M. Ritchie schrieben, stammt somit indirekt über B von BCPL ab.

6.3 Aussichten von C

Obwohl C als Systemprogrammiersprache entwickelt wurde, eignet sie sich auch für die meisten anderen Anwendungsgebiete. Die Programmierung in C unter UNIX bietet sich an, da die Programme aufgrund gleicher Sprachen besonders gut harmonieren. Aber auch unter anderen Betriebssystemen hat sich C zunehmend durchgesetzt. Microsoft programmiert fast ausschließlich in C und stellt seinen Compiler auch dem Nutzer zur Verfügung, womit sichergestellt ist, daß professionelle Entwicklungssysteme vorhanden sind. Auch Borland ist mit Turbo C und Turbo C++ dabei. Das C++ ist dabei die Bezeichnung für einen neuen Sprachstandard, der OOP enthält.

```
for (vp=ltv; vp->nr != 0; vp++) {
  if (vp->ueb.mo <= og && vp->ueb.mo >= ug) {
    printf("\n%5d %-20s", vp->nr, vp->bez);
    printf("\t%2d\t%10ld, -M", vp->ueb.mo, vp->pr);
    gp += vp->pr; } }
```

Programm 3: C

7 LISP

7.1 Namensklärung

LISP steht für LISt Processing – Listenverarbeitung. Man darf jedoch nicht annehmen, daß LISP für die Anforderungen im Buchmacher- oder Geschäftswesen konzipiert sei. Die Liste ist vielmehr eine besondere Variablenart der Sprache. LISP ist eine Sprache, die für alle Einsatzzwecke im Zusammenhang mit künstlicher Intelligenz (KI) prädestiniert ist.

7.2 Problem und erste Lösung

John McCarthy war mit den Formulierungs- und Notationsproblemen Ende der fünfziger Jahre unzufrieden. Sein Interesse an intelligenten Systemen weckte 1949 auf einem Symposium John von Neumann in ihm. Für das Projekt, ein Programm zum Beweis von geometrischen Aussagen aufzustellen, eigneten sich keine der damals vorhandenen Sprachen besonders gut. McCarthy interessierte die Listenstruktur in IPL und gefielen einige Eigenschaften von FORTRAN. Er versuchte, Listenverarbeitung in FORTRAN zu verwirklichen. So entstand FLPL.

7.3 Zufällige Entstehung

Doch die Formulierungsmittel von FORTRAN reichten bald nicht mehr, es mußte eine neue Programmiersprache entwickelt werden. Etwa um 1958 bekam das Gedankengebilde den Namen LISP. Die Definition der Sprache geschah eher unabsichtlich mit Hilfe einer Funktion EVAL, die dazu benutzt werden konnte, interpretativ Programme in LISP zu realisieren. Diese Funktion lief wahrscheinlich das erste Mal Ende 1958, gewiß aber Anfang 1959. 1960 gab McCarthy das LISP1 Manual heraus. Die Sprache enthielt viele Fehler, und so kam 1962 eine völlig revidierte Version von LISP heraus. Da LISP2 schon in Planung war, erhielt sie den Namen LISP1.5. Dieses System setzte sich schließlich durch und fand eine große Verbreitung, wohl auch deshalb, weil LISP2 noch lange auf sich warten ließ. 1967 erschien mit LISP1.6 eine leicht verbesserte Version.

```
(de length(l)
(prog(n) (setq n 0)
zykl(cond(l
(setq l(cdr l))
(setq n(add1 n))
(go zykl)))
(return n)))
```

7.4 Metasysteme und LISP2

In der Zwischenzeit entstanden Metasysteme, das heißt, es wurde auf der Grundlage von LISP versucht, Elemente anderer Sprachen einzufügen. Mit diesen Metasystemen erhalten viele Programmierer das erste Mal Kontakte mit LISP.

Programm 4: LISP

„Obwohl als Nachfolger ... gedacht, ist LISP2 doch hauptsächlich als Metasystem entstanden und als solches wieder untergegangen.“ /1/. Diese Version wurde mit Vorschußlorbeeren versehen. Doch sie erkrankte am „Feature-Krebs“ („Mehr, mehr, mehr ...!“). „Leider entwickelten sie es zu Tode ...“ /1/.

7.5 Zukunftschancen

Damit kamen Stimmen auf, die LISP bereits eine Grabrede hielten. Doch die Zeit war nur nicht reif für LISP. Heute gewinnt die Sprache immer mehr an Einfluß, denn sie ist mit PROLOG die einzig brauchbare Sprache für die Entwicklung der KI.

8 Kurzüberblick über andere Sprachen

8.1 PROLOG

PROLOG steht für PROgramming in LOGic – Programmieren in Logik. Sie ist eine deskriptive Programmiersprache, das heißt, dem System werden Sachverhalte beschrieben und Fragen gestellt, deren Antwort es dann versucht, aus den Sachverhalten zu ermitteln. PROLOG wurde 1972 in Marseille von Alain Colmerau's Forschungsteam erstmalig implementiert. Sie ist mit LISP die wichtigste Sprache für Probleme der KI und hat deshalb große Chancen in der Zukunft.

8.2 COBOL

Die COmmom Business Orientated Language – also etwa An allgemeinem Geschäft orientierte Sprache – wurde circa 1960 entwickelt unter der Vorgabe, eine Sprache zu entwickeln, die leicht erlernbar ist und im kommerziellen Bereich besondere Stärken zeigt. In COBOL programmiert man fast in natürlichsprachigem Englisch. Jedoch wird die Programmierung damit umständlicher. Mangels Besserem und aufgrund eines gewissen Beharrungsvermögens der Anwender hat sie sich jedoch durchgesetzt.

8.3 PL/1

Hochtrabend als Programmiersprache Nummer 1 bezeichnet sollte PL/1 sowohl FORTRAN als auch COBOL ersetzen. Doch die 1965 erschienene Sprache erfüllte die hohen Erwartungen nicht. Sie ähnelt entfernt Pascal, nur ist die strukturierte Programmierung nicht so ausgeprägt und die Ein-/Ausgabemöglichkeiten sind weiter gefaßt. PL/1 ist eine Sprache für Großrechner geblieben.

8.4 ALGOL

ALGOL (ALGOritmic Language – Algorithmische Sprache) ist 1956 bis 1962 von einem Komitee entwickelt worden, dem auch John McCarthy, der Entwickler von LISP, angehörte. Sie ist ebenso wie FORTRAN eine auf mathematische Sachverhalte spezialisierte Sprache, die weitreichendere und bessere Konzepte als FORTRAN hatte, sich gegen sie aber nicht durchsetzen konnte.

8.5 FORTH

Der Name der neu zu entwickelnden Sprache durfte nur fünf Buchstaben lang sein, deshalb wurde aus „fourth“ einfach das „u“ gestrichen. Es sollte angedeutet werden, daß es sich um eine Sprache für die 4. Rechnergeneration handelt. Man erreichte damit eine Doppelsinnigkeit, denn „forth“ heißt auch „vorwärts“. Dieses Ziel wurde nicht erreicht, denn FORTH ist fast wieder

verschwunden. FORTH fällt durch UPN und eine konsequente Benutzung des Stacks auf. Der Stack ist das wichtigste Programmiermittel.

8.6 Ada

Ada ist eine nach Ada Augusta Lovelace, dem ersten Programmierer der Welt, eine Frau übrigens, die Programme für die von Charles Babbage entwickelte programmierbare Rechenmaschine schrieb und dabei wichtige Prinzipien wie Schleife, Unterprogramm und bedingten Sprung erfand, benannte Sprache, die Anfang der achtziger Jahre im Auftrag des Pentagon entwickelt wurde. Der Aufbau gleicht dem von Pascal. Die Sprache ist jedoch durch viele Zusätze („Feature-Krebs“, siehe bei LISP) so umfangreich geworden, daß der Vorteil der leichten Erlernbarkeit beseitigt worden ist. Ada ist eine sehr mächtige Sprache, die aber voraussichtlich in nächster Zeit nur auf Großrechnern zu finden sein wird.

9 Fazit

Die Geschichte der Programmiersprachen ist wechselvoll und interessant wie die Geschichte der Menschen auch, was Wunder, ist sie doch direkt vom Menschen abhängig. Es wird interessant sein, die Entwicklungen der nächsten Zeit zu beobachten. Wird C ihren Siegeszug fortsetzen? Wird Pascal vielleicht von Modula verdrängt? Oder verschwinden alle Drei im Zuge der Entwicklung der KI? Denn keine Wissenschaft entwickelt sich so schnell, wie der Zug der Informatik jetzt fährt. Wichtig jedoch ist und wird sein, daß man die Großen der Informatik nicht vergißt, seien es „Softwarekünstler“ oder „Hardwarebastler“!

Sach- und Worterklärungen

Betriebssystem	Programm, das bei kleineren Computern noch im ROM liegt und die Zusammenarbeit zwischen Programmen und (direkt) Hardware oder (indirekt) BIOS ermöglicht.
BIOS	Basic Input Output System – Basis-Eingabe/Ausgabe-System. Ermöglicht Zusammenarbeit zwischen Hardware und Betriebssystem. Somit kann dasselbe Betriebssystem auf verschiedenen Rechnern laufen, das BIOS paßt die Eigenheiten an.
Compiler	Gegenstück zum Interpreter. Das gesamte Programm wird in einem Zug übersetzt. Das hat zum einen den Vorteil, daß das Programm später sehr schnell ausgeführt wird, zum anderen, daß alle Fehler in der Syntax sofort bemerkt werden. Damit kommen in compilierten Programmen Fehler seltener vor.
Hardware	Gesamtheit aller Geräte im weitesten Sinn.
HLL	High Level Language – höhere Programmiersprache (z. B. LISP)
Interpreter	Gegenstück zum Compiler. Ein Programm wird während der Laufzeit Zeile für Zeile untersucht, analysiert und übersetzt. Wird eine Zeile 100 Mal abgearbeitet, so muß sie ebensooft übersetzt werden, was natürlich zu Lasten der Abarbeitungsgeschwindigkeit fällt. Hat jedoch den Vorteil, daß Befehle im interaktiven Betrieb direkt nach Eingabe ausgeführt werden können, beispielsweise erscheint nach der Aufforderung $2^{(3+13)}$ sofort die Antwort 65536.
LLL	Low Level Language – niedere Programmiersprache (z. B. Assembler)
MLL	Medium Level Language – mittlere Programmiersprache. HLL, die sich an die Maschinensprache annähert (z. B. C)
Multitasking	task – Aufgabe. Beim Multitasking führt der Computer mehrere Programme gleichzeitig aus. Da dafür mehrere Prozessoren vorhanden sein müßten, werden einige Befehle des einen, dann einige des anderen Programmes abgearbeitet. Die Programme teilen sich also die Zeit des Prozessors, was man als Time-Sharing bezeichnet.
Multi-User-Multi-Task-System	Ein Betriebssystem, das mehreren Benutzern gleichzeitig einen (virtuellen) Rechner bietet und Multitasking beherrscht.
objektorientiert	Programmierstil. Die Teile des Programmes sind fest mit den Daten verbunden und bilden ein Objekt. Entspricht unserer natürlichen Denkweise: Wir verbinden mit „Fahrstuhl“ die Eigenschaften „fährt hoch“, „fährt hinunter“ usw.

problemorientiert	Programmierstil. Programmteile sind einzelne Problemlösungen und von den Daten getrennt. Abstrahierende Programmierweise.
OOP	Object Orientated Programming – objektorientierte Programmierung
Prozessor	Zentraler Bestandteil des Computers. Er verarbeitet direkt die Befehle in Maschinensprache und koordiniert die restlichen Baugruppen.
ROM	Read Only Memory – Nur-Lese-Speicher. Seine Informationen sind fest verdrahtet und bleiben ohne Spannungszufuhr erhalten. Das muß man jedoch mit der Unveränderbarkeit der Daten bezahlen.
Stack	Stapel oder Kellerspeicher. Meist nach dem LIFO (Last In First Out)- oder FILO-Prinzip organisiert. Einzelne Daten werden wie ein Stapel Akten übereinandergelegt. Das zuletzt aufgelegte Datum wird als erstes wieder heruntergenommen. Gut geeignet für UPN und als Zwischenspeicher bei rekursiven Operationen.
Strukturierung	Programmiertechnik, bei der ähnlich des Aufbaus eines (wissenschaftlichen) Buches zusammengehörende Befehle zusammengefaßt werden. Strukturierte Programme sind sehr übersichtlich. Fehler sind leicht zu finden, Veränderungen leicht durchzuführen.
UPN	Umgekehrt Polnische Notation. Eine besondere Schreibweise von mathematischen Ausdrücken, die Klammern überflüssig macht und besonders gut mit Hilfe eines LIFO-Stacks zusammenarbeitet. Aus $a*(b+c)$ wird $a b c + *$: Zuerst werden a,b,c auf den Stack gebracht, dann b und c addiert und das Ergebnis mit a multipliziert.

Literaturverzeichnis

- /1/ Dr. Herbert Stoyan:
LISP – Anwendungsgebiete, Grundbegriffe, Geschichte.
Akademie-Verlag, Berlin 1980
- /2/ Dr.-Ing. Ludwig Claßen, Dipl.-Math. Ulrich Oefler:
UNIX und C. Verlag Technik, Berlin 1987
- /3/ Doz. Dr. sc. tech. Uwe Petersohn, Dr. rer. nat. Ursula Hans, Dipl.-Ing. Jürgen Pitschke:
Micro-PROLOG. Eine Einführung in das Programmieren.
Fachbuchverlag, Leipzig 1988
- /4/ Dipl.-Math. Eckhard Schiller:
Computerwissen für alle. Fachbuchverlag, Leipzig 1988

- /5/ Kilian Keidel, Hans Joachim Müller:
 Informatik. Einführung Pascal. Ein Lehr- und Arbeitsbuch.
 Bayerischer Schulbuch-Verlag, München 1988

- /6/ Fragen an Turbo Pascal 6.0. In: Borland Zeitung. 1990, Dezember

- /7/ Prof. Dr. rer. nat. habil. Helmut Adler, Dr.-Ing. Hans-Ulrich Karl:
 FORTRAN 77. Eine Einführung in das Programmieren.
 Fachbuchverlag, Leipzig 1990

- /8/ Markus Zietlow, Herwig Weihe:
 BASIC im Wandel der Zeiten. In: DOS International. Das Magazin für aktive PC-Anwender, 5. Jahrgang. 1991, Nr. 2

- /9/ Dr. Gerhard Paulin, Dr. Hans Schiemangk:
 Programmieren mit Pascal. Akademie-Verlag, Berlin 1986

Beispielprogramme

1	FORTRAN	6
2	Pascal	7
3	C	9
4	LISP	10