Programmieren, Programmiersprachen *Basics*

In diesem Beitrag wird das Thema „Programmieren“ von zwei Seiten her beleuchtet: einerseits von der historischen Entwicklung her und andererseits von den verwendeten Werkzeugen her. Um den Text leichter lesbar zu machen, sind die Literaturangaben und Ergänzungen in den Fußnoten zu finden.

Bei einigen Programmiersprachen ist ein Beispiel zu finden: jene Zahlen zwischen 1 und 100, die nicht durch 3 teilbar sind, werden addiert. Dargestellt sind immer nur Ausschnitte aus kompletten Programmen.

Martin Weissenböck

Ein paar Grundbegriffe

Die folgenden Grundbegriffe sollen beim Verständnis dieses Beitrages helfen. Allerdings ist nicht beabsichtigt, alle modernen Verfahren aufzuzählen.

Spagetti-Code

Scherzhafte Bezeichnung für einen schlechten und veralteten Programmierstil, bei dem der Programmablauf häufig durch Sprunganweisungen unterbrochen wird. Sprünge, die im Programm kreuz und quer herumführen, machen ein Programm unleserlich. In weiterer Folge sind solche Programm schwer oder nicht wartbar und müssen bei Änderungen oft komplett ersetzt werden.

Äußeres Zeichen dieses Programmierstils ist die Verwendung von Flußdiagrammen oder Zeitablaufplänen. Zwar können auch mit Flußdiagramme gut strukturierte Programme beschrieben werden, aber erst die Verwendung von Struktogrammen erzwingt einen guten Programmierstil.

Strukturierte Programmierung

Der erste Ansatz, das Programmieren von einer „Trial-and-Error“- (Versuchs-Irrtums-) Methode zu einer ingenieurmäßigen Kunst zu führen, war die strukturierte Programmierung. Ausgangspunkt war ein Beitrag[1] des Informatikers Dijkstra. Er erläuterte seine Beobachtung erläuterte, daß die Verwendung von Sprunganweisungen („Goto-Statements“) Programme schlechter, das heißt unleserlich und nicht wartbar, macht.

Ältere Programmiersprachen und einfache Assemblersprachen kannten und kennen keine andere Form, den Programmablauf zu verändern. Zwar können auch in diesen Sprachen strukturierte Programme geschrieben werden, der Programmierer muß aber dabei mit großer Disziplin vorgehen.

Nach den Konzepten der strukturierten Programmierung[2] dürfen Programme nur aus folgenden funktionalen Blöcken bestehen: Anweisung und Unterprogrammaufruf, Ein- und Mehrfachverzweigung, Schleife. Jeder funktionale Block muß genau einen Eingang und einen Ausgang haben.

Sprünge, die direkt aus einem Block herausführen oder in einen Block hineinführen, sind verboten. Sprungbefehle werden in modernen und didaktisch wertvollen Programmiersprachen bestenfalls als allerletzter Rettungsanker eingesetzt (siehe z.B. Pascal); andere Sprachen haben den Sprungbefehl komplett gestrichen (z.B. Java).

Graphisch werden strukturierte Programme durch Struktogramme[3] dargestellt. Anders ausgedrückt: ist ein Programm durch ein Struktogramm darstellbar, dann ist es auch strukturiert programmiert.

Objektorientierte Programmierung

Die ständig steigenden Kosten der Softwareentwicklung und -wartung haben dazu geführt, daß neue Wege für die Wiederverwertung von Programmcode gesucht wurden.

Die objektorientierte Programmierung ist durch drei Kriterien gekennzeichnet:

- die Datenkapselung,
- die Vererbung und
- die Polymorphie.

Datenkapselung: Bestimmte Daten arbeiten oft nur mit bestimmten Routinen (Unterprogrammen) zusammen, ja die Programmlogik wird oft empfindlich gestört, wenn diese Daten von einem anderen Teil des Programms aus verändert oder auch nur gelesen werden können. Diszipliniertes Programmieren kann hier helfen; besser aber, wenn eine Programmiersprache selbst vor Fehlern schützt. Werden nun Daten samt ihren zugehörigen Routinen zu einer Einheit zusammengefaßt, die außerdem gegen unbefugten oder unabsichtlichen Zugriff von außen schützt, spricht man von einer Datenkapsel. Die Daten in der Kapsel heißen oft *Eigenschaften*, die Unterprogramme zu deren Bearbeitung *Methoden*. Die vereinbarten Kapseln werden meist Klassen, zur Verwirrung aber auch manchmal Objekte genannt; konkrete Ausprägungen heißen Instanzen. Der Unterschied zwischen Klassen und Instanzen ist leicht erklärt: das Kochrezept für einen Gugelhupf ist eine Handlungsanweisung, selbst aber im allgemeinen nicht zu genießen. Das Gugelhupfrezept ist eine Klasse(nvereinbarung). Erst der konkret gefertigte Gugelhupf (die Instanz) ist wirklich zum Essen geeignet. Auch mehrere Instanzen können nach einem Rezept gefertigt werden!

1 Dijkstra E.W.: Go To Statement Considered Harmful; Letter to the Editor; Communications of the ACM, März 1968.

2 Dahl O. J.; Dijkstra E. W.; Hoare C. A. R.: Structured Programming; Academic Press; London 1972.

3 Nassie-Shneidermann-Diagramme.

Vererbung: Aufgabenstellungen sind einander oft ähnlich. Wenn nun für ein konkretes Problem eine Lösung gefunden worden ist, liegt der Gedanke nahe, das bestehende Programm an die neue Aufgabe anzupassen. Allerdings sollten Änderungen und Eingriff in bestehende und ausgetestete Programme vermieden werden. Die Vererbung geht einen besseren Weg: bestehende Programme werden nicht verändert; neue Aufgaben werden durch neue Routinen umgesetzt. Um beim Gugelhupf-Beispiel zu bleiben: der bewährte Gugelhupf kann durch einen Gupf mit Schlagobers verbessert („ausgebaut“) werden, das Grundrezept wird vererbt. Ein Schokoladegugelhupf (mit einem Überzug aus Schokolade) wäre eine andere vererbte Klasse. Die neuen Klassen (Gugelhupf mit Schlagobers oder mit Schokolade) sind abgeleitete Klassen der Basisklasse „Gugelhupf“.

Polymorphie: Wenn eine Methode mitvererbt wird, soll sie auch in der abgeleiteten Klasse richtig funktionieren. Wenn eine Methode „weiß“, auf welche Art von Klasse sie anzuwenden ist, und dementsprechend reagiert, dann ist sie polymorph.

Ereignisgesteuerte Programme

Moderne graphische Benutzeroberflächen (wie zum Beispiel von Windows) bieten dem Benutzer viele Vorteile:

- Die Bedienung der Programme ist standardisiert. Wer ein Programm bedienen kann, findet sich rasch auch in einem anderen zurecht.
- Viele Aktionen können durch etliche, unterschiedliche Befehle ausgelöst werden. Der Benutzer hat die Chance, das Programm so zu bedienen, wie es ihm am besten gefällt. Als Beispiel:
- Anklicken eines Menüpunktes
- Wahl eines Menüpunktes über einen bestimmten, hervorgehobenen Buchstaben (oder ein bestimmtes Zeichen)
- Ansteuern eines Menüpunktes über die Cursortasten und Drücken der Return-Taste
- Shot-Cut, also eine bestimmte Tastenkombination
- Peripherieteile, wie zum Beispiel der Drucker, müssen nicht für jedes Programm separat installiert werden.
- All diese Vorteile verlangen nach einem neuen Programmierwerkzeug. Wenn der Benutzer praktisch zu jedem Zeitpunkt über die Maus irgendeinen Punkt anklicken kann oder auch auf x andere Arten einen Befehl geben kann, ist die übliche Abfolge eines Programms „Initialisierung - Eingabe - Verarbeitung - Ausgabe - Abschluß“ nur schwer einzuhalten. Statt des normalen sequentiellen Programmablaufes müssen Programme nun auf Ereignisse reagieren. Die ersten Versuche, mit traditionellen Programmwerkzeugen, wie etwa Unterprogrammsammlungen, Windows-Oberflächen zu programmieren, war bestenfalls Spezialisten vorbehalten.

Das neue Paradigma heißt „ereignisgesteuerte Programmierung“. Erst der sinnvolle Einsatz der objektorientierten Programmierung machte die breite Anwendung von Windows-Oberflächen möglich. Mit Visual Basic und Delphi können schließlich sehr bequem die Benutzeroberflächen erstellt werden. Mehr dazu im nächsten Abschnitt.

Didaktisches und historisches

Die Pionierzeit

Die Zeit bis 1980:

Programmieren ist eine geheime Kunst: Programme werden mit Lochkarten und Lochstreifen, manchmal auch mit Markierungskarten erstellt. Die Fähigkeit, Lochkarten und Lochstreifen erstellen und vor allem korrigieren zu können ist fast wichtiger als die Kenntnisse der Grundbegriffe der Informatik. In den Schulen dominieren exotische Systeme, Time-Sharing-Anschlüsse und einige Anlagen, die damals als mittlere bis große Anlagen gegolten haben. Ende der 70er Jahre tauchen die ersten Kleinrechner (PET, Apple II, ...) auf.

Programme werden vor allem in Basic erstellt. Liebevoll werden die unterschiedlichsten Dialekte gepflegt: Applesoft-Basic, Integer-Basic, GW-Basic, True-Basic, Microsoft-Basic und wie sie alle geheißen haben. Assembler-Programmierung wird von einigen als der einzige wahre Weg zur Programmierkunst angesehen. Auch Fortran-Programme finden ihren Weg von den Universitäten an die Schulen.

Theoretische Überlegungen zur Programmiererei waren nicht weit verbreitet; das wichtigste Ziel war ein funktionierendes Programm. Zwar wurden zur Dokumentation Flußdiagramme eingesetzt, aber trotzdem oder vielleicht sogar deshalb waren die Programme äußerst unübersichtlich und schwer zu warten.

Diese „Programmiertechnik“ hatte auch schwerwiegende wirtschaftliche Konsequenzen. Verließ ein Programmierer eine Firma, mußten nicht selten alle seine hinterlassenen Programme mangels Dokumentation neu geschrieben werden.

Im Unterricht dominierten mathematische Probleme. Die EDV wurde vielfach als Fortsetzung der Mathematik mit anderen Mitteln angesehen, ohne aber dabei auf die speziellen Problemlösungsmöglichkeiten der EDV (wie zum Beispiel eine sukzessive Approximation) wirklich im Detail einzugehen.

Das goldene Zeitalter

Die 80er Jahre

Neue Konzepte beginnen in den Schulen Fuß zu fassen. Religionskriegsartige Auseinandersetzungen und Diskussionen um die richtige Programmiersprache werden geführt. Die Idee der strukturierten Programmierung beginnt sich an den Schulen durchzusetzen. Sprachen, die dieses Konzept fördern, werden eingesetzt: Pascal ist wegen seiner didaktischen Vorteile sehr beliebt. Trotzdem verharren einige beim geliebten Spagetticode der Basic-Zeit.

Inhaltlich erlebt die Programmierkunst an den Schule ihre Blütezeit: auch komplizierte Konstrukte, wie Zeiger, Records und Dateibearbeitung werden gelehrt und gelernt. Systeme mit integrierter Entwicklungsumgebung und eingebauten Testhilfen (Debugging) erfreuen sich großer Beliebtheit. Turbo-Pascal wird allgemein eingesetzt.

Endlich werden auch nicht-mathematische Probleme in Programme umgesetzt. Zeichenverarbeitung und vor allem die Graphik bringen neue Programmideen.

Gegen Ende der 80er Jahre bewegt ein neues Paradigma die Informatik: die objektorientierte Programmierung. Mit dem Ausspruch „Meine Katze ist objektorientiert“ wird dieser Trend in Artikeln persifliert. Natürlich erweckt dies auch das Interesse der Lehrer: die Lehrinhalte werden überprüft, ob sie zu dem neuen Trend passen.

Turbo-Pascal wird zur objektorientierten Sprache. C++ ist voll im Trend der Zeit.

Die Gegenwart

Die 90er Jahre

Graphische Benutzeroberflächen gehören zum Standard. Leider halten vorerst die Programmierwerkzeuge nicht Schritt. Spezialisten müssen mit „Entwicklungssystemen“ für Windows-Oberflächen mit mehreren hundert Unterprogrammen und einer unüber-

schaubaren Zahl an Parametern arbeiten, für die Schule sind diese Werkzeuge ungeeignet. Dazu kommt noch, daß ereignisgesteuerte Programme ein Umdenken erforderlich machen.

Auch Versuche, wie *Pascal für Windows* oder Formulargeneratoren sind nicht sehr erfolgreich. C++ bringt Hilfsprogramme (Wizzards), aber die entstehenden Kilobyte an Sourcecode sind im Unterricht auch nicht gut handhabbar.

Erst Visual Basic von Microsoft bringt eine Trendwende. In einer ansprechenden graphischen Umgebung können Benutzeroberflächen gleich am Schirm konstruiert werden. Parallel dazu entsteht der Rumpf eines Programmes in Form von Unterprogrammvereinbarungen, die noch mit entsprechendem Programmcode gefüllt werden müssen.

Der Informatik-Unterricht wird mit immer neuen Inhalten bedacht: Tabellenkalkulationsprogramme und Datenbankprogramme ersetzen in vielen Fällen die hohe Programmierkunst. Grundkenntnisse der Textverarbeitung sollen vermittelt werden und die Telekommunikation verlangt ihren Platz im Unterricht. Bei gleichbleibender Stundenzahl und teilweiser Verlagerung in die 1. und 2. Jahrgänge der HTL sind inhaltliche Änderungen unausweichlich. der Anteil der Programmiererei im Unterricht geht zurück.

Die Zukunft

Der Beginn eines neuen Jahrzehnts, Jahrhunderts, Jahrtausends...

Die Programmierkunst reduziert sich im normalen Unterricht auf die Grundlagen, wie Anweisung, Verzweigung, Schleife und Unterprogrammaufruf. Programme treten vor allem als Makroersatz in Anwenderprogrammen auf.

Tabellenkalkulation und Datenbankprogramme werden durch kleine Programmsequenzen an die Bedürfnisse der Nutzer angepaßt.

Schüler bringen die Grundkenntnisse von der Vorläuferschulen mit. Bald nach der Jahrtausendwende werden alle HTL-Schüler schon im 1. Jahrgang mit einem Lap-Top ausgerüstet. Der Umgang mit dem Computer wird immer selbstverständlicher. Er ersetzt Heft und Taschenrechner.

Einige Abteilungen benötigen nach wie vor tiefere Kenntnisse: so ist zur Programmierung von Mikroprozessoren nach wie vor ein tieferes Programmierverständnis in den Abteilungen für Technische Informatik notwendig. Werden Meßdaten erfaßt, muß häufig zwischen Meßgerät und Auswerteprogramm noch ein kleines Erfassungsprogramm zwischengeschaltet werden; somit bleibt beispielsweise auch in den Elektrotechnikabteilungen dieses Segment erhalten, wandert aber eher in andere Unterrichtsgegenstände, wie Prozeßdatentechnik.

Programme sind nach wie vor wichtig im Bereich der Telekommunikation: animierte Internetseiten gehören zum guten Ton und diese Programme werden in Java, Javascript, Visual Basic und anderen Sprachen geschrieben. Nach und nach übernehmen aber auch Hilfsprogramme (ActiveX-Controls) diese Aufgabe.

Schließlich müssen die Internet- und Intranet-Server neue Aufgaben zur Aufbereitung von Daten übernehmen. Auch dazu sind Programmierkenntnisse notwendig. Aber auch hier übernehmen fertig konfigurierte Datenbankprogramme bald die Aufgaben des Programmierers.

Programmiersprachen

Ada

Das amerikanische Verteidigungsministerium, das „Departement of Defense“ (DoD), ist weltweit der wichtigste Käufer von Software. Kein Wunder, daß dieser Kunde größten Wert darauf legt, daß die gekaufte Software auf jeder Hardware eingesetzt werden kann. Da viele weit verbreitete Programmiersprachen herstellerspezifischen Wildwuchs aufweisen, wollte das DoD eine einheitliche Sprache - ohne Varianten, Versionen usw. Ada war nicht der erste Versuch: auch Algol 68 sollte universell sein und Java versucht es in der Gegenwart. Algol 68 war zu umfangreich; der Erfolg von Java ist zu beobachten.

Ada geht auf das Jahr 1974 zurück. Das DoD gab einen Auftrag zur Untersuchung der verwendeten Sprachen. In mehreren Schritten wurde eine neue Sprache entwickelt, die im Jahr 1979 vorgestellt wurde und zu Ehren von Augusta Ada Byron, Gräfin von Lovelake (1816-1851) Ada[4] genannt wurde.

Die Sprache Ada ist tatsächlich einheitlich geblieben. Jedoch fehlen alle modernen Ansprüche, wie die der objektorientierten Programmierung. Ada wird manchmal höflicherweise auch als objektbasierte Sprache bezeichnet; jedoch ist keines der drei Kriterien der objektorientierten Programmierung vorhanden.

Ada ist inzwischen weiterentwickelt worden.

Beispiel:

```
declare
  i: integer;
  sum: integer := 0;
....
for i in 1..100 loop
  if i mod 3 /= 0 then
    sum:=sum+i;
  end if;
end loop;
```

Algol 60

Algol steht für „**Algo**rithmic **L**anguage“, also „algorithmische Sprache“. Ende der 50er Jahre wurde erstmalig der Versuch gemacht, eine Programmiersprache zu entwickeln, die zur Darstellung und zur Programmierung von Rechenvorschriften (Algorithmen) geeignet sein sollte. Die Grammatik der Sprache, die Syntax, wurde durch ein eigenes Regelwerk beschrieben.

Algol 60[5] kennt nur numerische Datentypen. Fragen der Genauigkeit der Zahlendarstellung spielen keine Rolle. Der größte Nachteil: Algol enthält keine praktischen Befehle zur Ein- und Ausgabe von Daten.

Algol war für die Entwicklung moderner Sprachen sehr wichtig, hat aber heute keine Bedeutung mehr.

Beispiel:

```
INTEGER i, sum;
...
sum := 0;
FOR i:=1 STEP 1 UNTIL 100 DO
  IF i/3*3i THEN sum := sum + i;
```

Algol 68

Algol 68 war einerseits als Weiterentwicklung von Algol 60 gedacht, andererseits wurden völlig neue Konzepte erarbeitet. Algol 68 sollte die universelle Sprache sein - so gut wie alle damals aktuellen Ideen der Informatik wurden in der Algol-Report hineingeschrieben. Leider bleiben Fragen der Realisierung zurück: die Anforderungen an die Sprache waren so hoch, daß sie nie vollständig implementiert wurde.

4 ADA Programming Language: Military Standard MIL-STD-1815, 1980. United States of America, Department of Defense.

5 Revised Report on the Algorithmic Language ALGOL 60, Num. Math. 4, 420 (1963); Comm of the ACM 6, 1 (1963); The Computer Journal 5, 349 (1963).

Böse Zungen behaupteten, daß schon für das Lesen des Algol-68-Repots ein Informatik-Studium notwendig sei.

Auch Algol 68 hat heute nur theoretische Bedeutung.

APL

A Programming **L**anguage. (So einfach können Namen sein!) Die Sprache wurde von K. E. Iverson 1962 definiert und von A.D. Falkoff und L.M. Breed weiterentwickelt. APL ist eine Sprache mit einem völlig anderen Konzept: statt einer Sammlung von Unterprogrammen werden Operatoren für alle Befehle verwendet. Typisch für APL: eine eigene Tastatur und früher auch eigene Kugelköpfe für die Schreibmaschinen.

APL-Programme sind wie Stenogramme: kurz, prägnant und nur für Spezialisten zu lesen.

Da Vektoren und Matrizen Rechenobjekte sind, können Multiplikation, Invertierung, Transponierung usw. mit einem einzigen Operator ausgedrückt werden.

Assembler

Eine Gruppe von Programmiersprachen, die auf unterster Ebene die maschinennahe Programmierung erlauben. Assemblersprachen sind deshalb im allgemeinen rechner- bzw. prozessorspezifisch, Versuche mit „Universalassemblersprachen“ haben sich nicht besonders bewährt.

Basic

Beginners **a**ll-purpose **s**ymbolic **i**nstruction **c**ode. Die Sprache wurde Anfang der sechziger Jahre im Dartmouth College in den USA unter der Leitung von John G. Kemeny und Thomas E. Kurtz speziell für den Dialogbetrieb entwickelt. Die ersten Basic-Varianten waren fernab jeder modernen Softwareentwurfstechnik, das erste Beispiel zeigt ein derartiges Programmstück.

Basic wurde immer weiterentwickelt: Quick-Basic[6], Turbo-Basic[7], PowerBasic und viele andere Varianten erlaubten strukturierte Programme. Das zweite Beispiel soll einen Eindruck davon vermitteln.

Visual Basic ist nun ein wertvolles Programmierwerkzeug, vor allem für graphische Benutzeroberflächen. Mit den ersten Basic-Varianten hat es nur wenig gemeinsam.

Beispiel 1:

```
100  SUM = 0
110  FOR I=1 TO 100
120    ZW = INT (I/3)
130    IF I = ZW*3 THEN 150
140    SUM = SUM + I
150  NEXT I
```

Beispiel 2:

```
sum% = 0
FOR i% = 1 TO 100
  IF i% MOD 3  0 THEN
    sum% = sum% + i%
  END IF
NEXT
```

C

Die ersten Vorläufer von C[8] stammen aus 1970. C wurde ursprünglich zum Entwurf des UNIX-Betriebssystems entwickelt. C[9] ist heute eine wichtige Sprache bei der Programmierung von Mikrocomputersystemen.

Leider müssen in C schon für einfache Aufgaben (zum Beispiel das Einlesen von Zahlen) recht komplexe Sprachelemente (zum Beispiel Pointer) verwendet werden. C++ ist didaktisch besser geeignet.

C erlaubt sehr kompakte Programme, die dann aber auch schwer nachvollziehbar bzw. durchschaubar werden.

Beispiel:

```
int i, sum=0;
for (i=1; i<=100; i++)
  if (i % 3 != 0) sum += i;
```

C++

Diese Nachfolgesprache von C unter Berücksichtigung der objektorientierten Programmierung. Der Name kommt von dem „++“-Operator: dieser erhöht den Inhalt einer Variablen um 1. „C++“ ist somit ein erhöhtes (verbessertes) „C“. C++ hat alle Eigenschaften einer objektorientierten Sprache[10],[11]:

- Der wichtigste Begriff ist die *Klasse*. In Klassen („class“) werden Eigenschaften und Methoden zusammengefaßt.
- Die Vererbung ist nicht nur in einfach absteigender Folge möglich; vielmehr kann eine neue Klasse von mehreren anderen Klassen erben (Mehrfachvererbung). Allerdings bereitet gerade diese Eigenschaft einige Schwierigkeiten, da sie viel Aufwand beim Compilieren mit sich bringt.
- Methode können durch den Zusatz virtual zu virtuellen Methoden gemacht werden; damit werden abgeleitete Klassen polymorph.
- Darüber hinaus erlaubt C++ die Neudefinition (fast) aller vorhandenen Operatoren.

Mit Templates (Schablonen) werden Programme so geschrieben, daß sie einfach mit unterschiedlichsten Datentypen arbeiten können.

Schließlich enthält C++ einen sehr ausgeklügelten Mechanismus zu Behandlung von Fehlern, die während der Laufzeit auftreten. Mit try werden Anweisungen „probeweise“ ausgeführt. Wenn ein Fehler auftritt, kann dieser Fehler im Detail in einem catch-Block bearbeitet werden.

C++ bietet viele Möglichkeiten zur Programmierung von graphischen Benutzeroberflächen. (Visual Basic und Delphi sind in mancher Hinsicht praktischer. Mit der Version „C++ Builder“ von Borland werden die Vorteile der C++-Programmierung mit der von Delphi bekannten graphischen Gestaltung der Benutzeroberfläche kombiniert.)

---

6 Weissenböck, M.: Quick-Basic: ADIM-Band 49. Microsoft, April 1994.

7 Weissenböck M.: Turbo/Power Basic: ADIM-Band 41. Basic-Versionen 1 bis 3 von Power Basic Inc. Oktober 1995. Auch als Fachbuch über die Schulbuchaktion zu beziehen.

8 Weissenböck M.: Turbo-C. ADIM-Band 40. September 1997. Auch als Fachbuch über die Schulbuchaktion zu beziehen.

9 Eines der wichtigsten ersten C-Bücher: Kernighan B., Ritchie D.: The C Programming Language, New Jersey 1978.

10 Weissenböck M.: C++. ADIM-Band 50. September 1997. Auch als Fachbuch über die Schulbuchaktion zu beziehen.

11 Die klassische Beschreibung von C++ stammt vom Entwickler der Sprache selbst: Stroustrup B.: Die C++ Programmiersprache, Addison-Wesley-Verlag, 1987. Inzwischen ist der Markt voll von C++-Büchern.

Beispiel:

```
int sum=0;
for (int i=1; i<=100; i++)
  if (i % 3) sum += i;
```

Cobol

Common Business Oriented Language. Die älteste Programmiersprache für den kommerziellen Bereich. Die Entwicklung begann bereits 1959. Die erste standardisierte Fassung wurde 1964 als „USA Standard-COBOL" veröffentlicht.

Eine Grundidee von Cobol: durch eine an das Englische angelehnte Notation sollten die Programme „lesbar" werden (und vielleicht auch leichter zu schreiben sein).

Ein Beispiel: `ADD I TO SUM`

Cobol-Programme bestehen aus vier Teilen:

- Identification division: Angaben über Programmierer, Projekt usw
- Environment division: Angaben über die Hardware des Computers, Rechnertyp, Compiler, Dateien
- Data division: Datenteil, Vereinbarung von Variablen
- Procedure division: der Ausführungsteil

Cobol-Programme enthalten viel Programmtext!

---

dBase

dBase ist eigentlich keine Programmiersprache, sondern ein Datenbankprogramm. Allerdings können in dBase auch Funktionen automatisiert ablaufen: dazu dient diese Sprache.

Delphi (Pascal)

Ein Entwicklungssystem von Borland für die graphische Benutzeroberfläche Windows. Die Funktionalität entspricht der von Visual Basic, die zugrunde liegende Sprache ist Pascal.

Ab Delphi 2 wird eine spezielle objektorientierte Version von Pascal verwendet.

Forth

Die ersten Arbeiten an der Sprache Forth reichen bis in das Jahr 1969 zurück: Charles H. Moore hat am National Astronomy Observatory in Charlottesville (USA) Forth als Dialogsystem entwickelt, vor allem zur Lösung von numerischen Aufgaben. Der Name ist eine Verballhornung des englischen Wortes „**fo**u**rth**", als Hinweis auf die vierte Generation von Sprachen.

Die Eingabe erinnert stark an die Eingabe von Hewlett Packard Taschenrechnern.

Die Grundelemente der strukturierten Programmierung sind auch in Forth zu finden.

Fortran

Formula **Tran**slator. Die älteste höhere Programmiersprache für den technisch-wissenschaftlichen Bereich wurde 1955 im wesentlichen von der Firma IBM mit dem Ziel entwickelt, an Stelle der mühsamen Programmierung im Maschinencode Rechenoperationen und mathematische Formeln in einer Notation schreiben zu können, die der der Mathematik sehr ähnlich ist.

Aus den ursprünglichen Konzepten entstand Fortran II (1958) und Fortran IV (1962). 1964 wurde die Sprache von der amerikanischen Normenbehörde standardisiert.

Mit den Elementen der strukturierten Programmierung wurde daraus Fortran 77.

Beispiel:

```
      DO 1 I=1,100
      IF (I/3*3 .EQ. I) THEN
        SUM = SUM + I
1     ENDIF
```

Fortran 90

Der letzte Stand der ältesten höheren Sprache. Nach Fortran 77 ist wurden u.a. Datenstrukturen, abgeleitete Datentypen, Module und Pointervariablen aufgenommen. Fortran 77-Programme müssen auch unter Fortran 90 laufen.

Beispiel:

```
DO i=1,100
  IF (MODULO(i,3).NE.0)
    sum = sum + i
  END IF
END DO
```

J

J ist eine äußerst ungewöhnliche Sprache. Schon von der Diktion her: die Bestandteile und Befehle der Sprache werden Hauptwörter, Zeitwörter, Adverben usw. genannt. J erlaubt auch besonders kompakte Programme: Vektoren, Matrizen und auch höhere Objekte werden wie einfache Skalare verknüpft und behandelt. APL kann als Vorgängersprache angesehen werden, J geht aber weiter über die APL-Ideen hinaus.

Java

Ursprünglich ist Java entwickelt worden um einfache Systeme der Konsumelektronik steuern zu können. Java, eine Entwicklung von Sun, ist heute eine wichtige Sprache fürs Internet. Seiten mit Java-Elementen können auf allen Systemen, unabhängig von bestimmten Prozessoren und Betriebssystemen dargestellt werden, sofern nur ein passender Browser (Netscape, Internet Explorer) existiert. Damit können aber auch andere Programme diese „Plattformunabhängigkeit" nutzen. So kann beispielsweise ein Textverarbeitungsprogramm einmal entwickelt werden und sofort auf den verschiedensten Rechner eingesetzt werden.

Java[12] ist den Sprachen C und C++ in vielen Punkten ähnlich, hat aber auch Ideen aus anderen Sprachen übernommen. Java ist eine objektorientierte Sprache.

Javaprogramme können eigenständig ablaufen oder von Browser-Programmen abgearbeitet werden. Ursprünglich waren nur relativ langsame Interpreter erhältlich, aber inzwischen werden Javaprogramme, die übers Internet geladen werden, sofort („just-in-time") compiliert und damit signifikant schneller.

Microsofts Java-Implementierung und Entwicklungssystem heißt J++. Das hat nichts mit J zu tun, sondern soll nur die enge Verwandtschaft zwischen C++ und Java ausdrücken.

Javascript

Javascript klingt zwar wie Java, ist aber eine eigenständige Entwicklung. Javascript-Programme werden nur per Interpreter abgearbeitet. Damit ist der Ablauf relativ langsam, aber auch sehr flexibel. Javascript eignet sich daher eher für kleinere Programme.

Javascript enthält objektorientierte Elemente, sollte aber eher als objektbasiert bezeichnet werden.

---

12 Java-Kurse gibt's im Internet!

Lisp

List **p**rocessing. Eine Sprache zur Listenverarbeitung, von 1959 bis 1963 am MIT entwickelt. Lisp wird als AutoLisp[13] zur Programmierung von automatischen Abläufen in AutoCAD verwendet.

Logo

Eine Sprache für Einsteiger, graphisch orientiert, mit Konzepten der Listenverarbeitung. Logo wurde 1967/68 entwickelt. Vor allem wurde das „Turtle[14]"-Konzept bekannt: eine graphische Schildkröte wird durch einfache Befehle gesteuert. Da Logo somit sehr einfach zu erlernen ist und wegen der graphischen Darstellungen auch rasch Erfolgserlebnisse vermittelt, wurde es Jahre hindurch sowohl in der Lehrer-aus- und -weiterbildung als auch im Unterricht verwendet[15].

Modula

Modula stammt aus dem Jahr 1982 und ist Wirths Weiterentwicklung[16] von Pascal. Eigentlich wird immer Modula-2 gemeint, wenn von Modula die Rede ist. Die Module von Modula erlauben die Complierung von Programmen in Form von unabhängigen Bausteinen, da die Schnittstellen genau definiert sind.

Modula ist keine objektorientierte Sprache. Von der Firma Top Speed wird eine objektorientierte Version angeboten.

Der Nachfolger von Modula - wieder eine Entwicklung von Niklaus Wirth - heißt Oberon. Auch Modula-3[17] ist ein Nachfolger.

Beispiel:

```
VAR
  i,sum: INTEGER;
...
sum:=0;
FOR i:=1 TO 100 DO
  IF (i MOD 3)  0 THEN
    INC(sum,i)
  END
END;
```

Oberon

Oberon ist das jüngste Kind von Niklaus Wirth[18] und die Weiterentwicklung von Modula-2. Eigentlich ist die derzeitige Version auch schon Oberon-2[19]. Die Sprache steht als Public Domain Programm allen Interessenten kostenlos zur Verfügung. Die Besonderheit: zu Oberon gehört auch ein komplettes Entwicklungssystem[20], das zusammen mit Oberon auf einer einzigen Diskette Platz findet.

Oberon läuft auch unter Windows. Oberon ist eine objektorientierte Sprache[21].

Beispiel:

```
VAR
  i,sum: INTEGER;
...
sum:=0;
FOR i:=1 TO 100 DO
  IF (i MOD 3)  0 THEN
    INC(sum,i)
  END
END;
```

Occam

Occam ist eine Sprache zur Programmierung paralleler Prozesse. Der Name leitet sich von William von Ockham[22], einem Mönch aus dem 14. Jahrhundert in England, ab. Occam enthält nur fünf Grundprinzipien, nämlich die Sequenz, die Parallelität, die Alternative, die Bedingung und die Schleife.

Pascal

Pascal wurde an der eidgenössischen Technischen Hochschule in Zürich von Professor Nikolaus Wirth als Unterrichtssprache erfunden, im Jahr 1971 veröffentlicht[23] und an der University of California San Diego zum UCSD-Pascal weiterentwickelt. Als Turbo-Pascal von Borland war es eine Zeit lang die schnellste und billigste Hochsprache für Kleinrechner.

Mehrere Implementierungen haben daraus eine objektorientierte Sprache gemacht; die Version von Borland hat sich durchgesetzt.

Aus Pascal entstand Modula.

Der Nachfolger von Turbo Pascal war als Pascal für Windows vorerst wenig erfolgreich. Erst Delphi brachte für Borland den Erfolg bei der ereignisgesteuerten Programmierung.

Beispiel:

```
var
  i, sum: integer;
....
sum:=0;
for i:=1 to 100 do
  if (i mod 3)  0 then sum:=sum+i;
```

Perl

Practical **E**xtraction and **R**eport **L**anguage. Eine C-ähnliche Sprache, die vor allem auf Unix-Rechnern[24] verwendet wird, obwohl Perl-Implementierungen nun schon für fast alle Plattformen vorliegen. Die Stärke von Perl ist die Analyse und Modifikation von Texten, vor allem durch sogenannte reguläre Ausdrücke.

Perl spielt eine große Rolle, wenn Unix-Rechner als Internetserver eingesetzt werden.

Inzwischen ist Perl bei der Version 5[25] angelangt: dieses Perl ist bereits objektorientiert.

---

13 Karl Habenicht, ADIM-Band 54, auch als Fachbuch über die Schulbuchaktion erhältlich.

14 Turtle = Schildkröte

15 Stormer W., Zierler P.: Logo. ADIM-Band 36. Logo-Version der Firma IBM. 2. Auflage, 1988.

16 Wirth N.: Programming in Modula-2. Springer-Verlag, 1982.

17 Harbison S.: Modula-3. Prentice-Hall, Inc., 1992.

18 Wirth N., Gutknecht J.: The Oberon System. Software-Practice and Experience, 19: 857-893, 1989.

19 Mössenböck H., Wirth N.: The Programming Language Oberon-2. Structured Programming, 12(4): 179-195, 1991.

20 Reiser M.: The Oberon System, User Guide and Programmer's Manual. Addison Wesley, 1991.

21 Mössenböck H.: Objektorientierte Programmierung in Oberon-2. Structured Programming, 12(4): 179-195, 1991.

Beispiel:

```
for ($i=1; $i <= 100; $i++)
{ if ($i % 3 != 0)
   { %sum += %i;  # Die geschwungenen Klammern
   }              # sind notwendig
}
```

PL/I

Programming Language One, eine Entwicklung der IBM. (Übrigens im Wiener Labor der IBM unter der Leitung von Prof. Zemanek entwickelt.) PL/I enthält einerseits Elemente von Algol, Fortran und Cobol, andererseits aber auch (zum Zeitpunkt der Entwicklung) völlig neue Konzepte, wie zum Beispiel das Arbeiten mit Zeigern.

PL/I[26] ist eine sehr umfangreiche und mächtige Sprache, die letztenendes auch viele Ressourcen bindet. Inzwischen gibt es bereits PL/I-Compiler, die unter OS/2 auf einem PC laufen.

Beispiel:

```
declare i,sum binary fixed;
sum = 0;
do i=1 to 100;
  if mod(i,3) = 0 then
  sum = sum + i;
end;
```

PL/M

Klingt wie PL/I und ist eine Programmiersprache für Mikroprozessoren.

Prolog

Bei „herkömmlichen“ Sprachen muß zum Erstellen eines Programms ein Problem analysiert und dargestellt werden; danach wird ein Algorithmus gesucht, der dann wieder programmiert wird. Bei Prolog-Programme fällt die Suche nach einem Algorithmus weg, da die Problembeschreibung direkt in ein Programm umgesetzt wird. Die Sachverhalte werden in Form von *rules* (Regeln) und *facts* (Fakten) beschrieben, ein *goal* ist das Ziel der Berechnung. Wie das Problem gelöst wird, bleibt dem Rechner überlassen.

Prolog-Programme eignen sich besonders für bestimmte nicht-numerische Aufgaben: etwa das Suchen nach einer optimalen Fahrtroute oder nach allen Lösungen eines logischen Problems.

Visual Basic

Basic hat im Laufe seiner Geschichte etliche Wandlungen durchgemacht. Visual Basic[27] war die erste Sprache, die einen *bequemen* Zugang zur graphischen Benutzeroberfläche von Windows erlaubt hat.

Visual Basic ist außerdem für die Office-Pakete von Microsoft wichtig: sollen Vorgänge automatisiert werden, waren bisher sogenannte Makros - unterschiedlich in jedem der genannten Programme - notwendig. Nun wird Visual Basic for Applications als gemeinsame Programmiersprache für alle automatisierten Abläufe in den Office-Paketen benutzt.

Eine weitere Variante, Visual Basic Script, ist Microsofts Antwort auf Javascript: auch mit VBScript (so die Kurzbezeichnung) können Internetseiten animiert werden.

Quellen

Das Internet ist voll Freeware- und Sharewareprodukten. Unix mit allen seinen Derivaten spielt dabei eine ganz besondere Rolle, da dafür besonders viele Sprachsysteme angeboten werden. Interessenten werden eingeladen, nach GNU-Sammlungen Ausschau zu halten.

Eine gut gewartete Sammlung von Programmen aller Art findet sich an der TU Wien unter *http://gd.tuwien.ac.at*.

Eine umfangreiche Kollektion von verschiedenen Programmiersprachen auf zwei CDs ist unter dem Titel „Power-Programmierung“ vom Verlag tewi herausgekommen (Bestell–Nummer 62629)

Software zu Windows-NT

☞Das Promotion Paket “**get connected –Schulen ans Intenet**” kann von jeder Schule bei einem EDU-Select Händler bezogen werden.

☞**Resource Kit**: Ist ein Buch und/oder CD in dem Microsoft wertvolle Utilities rund um Windows NT Server verlegt. Derzeit gibt es zu NT Server zwei Resource Kits die beide im Verlag Microsoft Press erschienen und im gut sortierten Buchhandel erhältlich sind.

☞**TechNet**: Ein Abo, das regelmäßig CDs für die Supportarbeit liefert. Näheres erfahren sie unter *http://www.eu.microsoft.com/technet*.

User Agent: Ein Dienstprogramm für die automatische Anlage vieler Benutzerkonten und deren Verwaltung. Näheres unter *http://www.edu-heldpesk.ac.at/uagent.htm*.

22 Pountain D., Rudolph R.: Occam, Das Handbuch. Heise, Hannover 1987.

23 Wirth N.: The Programming Language Pascal. Acta Informatica, I: 35-63, 1971.

24 Empfehlenswert: Schwartz R.: Learning Perl. O'Reilly & Associates, Inc. 1993.

25 Zum Einsatz in Webservers siehe: Middleton B. u.a.: Web-Programmierung mit Perl 5, Markt & Technik, 1997.

26 programming language PL/I: American National Standard ANSI X3.53-1976

27 Köberl H., Podenstorfer E., Zahler Ch.:Visual Basic. ADIM-Band 61. September 1997