

Zeitschrift: Vermessung, Photogrammetrie, Kulturtechnik : VPK = Mensuration, photogrammétrie, génie rural

Herausgeber: Schweizerischer Verein für Vermessung und Kulturtechnik (SVVK) = Société suisse des mensurations et améliorations foncières (SSMAF)

Band: 85 (1987)

Heft: 10

Rubrik: Informatik = Informatique

Autor: [s.n.]

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. [Mehr erfahren](#)

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. [En savoir plus](#)

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. [Find out more](#)

Download PDF: 05.07.2025

ETH-Bibliothek Zürich, E-Periodica, <https://www.e-periodica.ch>

Modell sind zusätzlich COGO Funktionen gespeichert. Das REC Modul ist austauschbar: pro Modul können ca. 500 Datenblöcke in aufgabengerechtem, frei wählbarem Format gespeichert werden.

Über eine Anschlussbuchse im festen Unterteil lässt sich das Instrument auch on-line an einen Computer anschliessen, mit einem Datenterminal Wild GRE3 bzw. GRE4 kombinieren oder mit Extern-Batterien verbinden.

Flexible, hochleistungsfähige Vermessungsstation

Die genaue Winkelmessung ist jedoch nicht der einzige Zweck dieses Instruments. Der Wild T1600 wird zu einer wirklich universellen Vermessungsstation, wenn man einen Infrarot-Distanzmesser Distomat aufsetzt. Mit den Modellen DI1000, DI5S, DI2000, DI3000 und DI 3002 bildet er ohne jegliche Kabelverbindung kompakte elektronische Tachymeter höchster Leistungsstärke, die mit Millimetergenauigkeit je nach gewähltem Distomat-Modell bis zu 14 km weit messen.

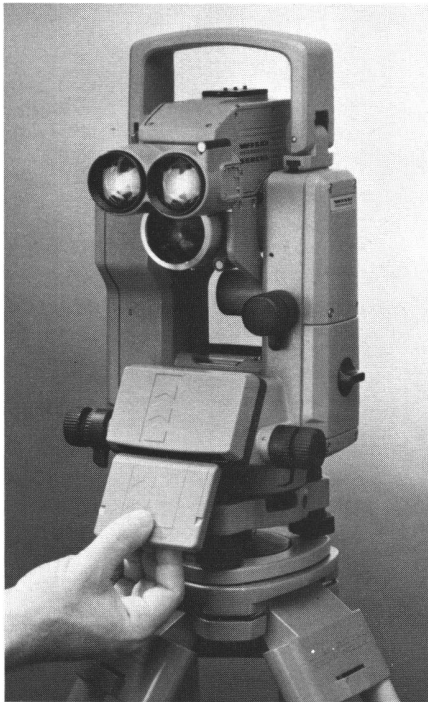


Abb. 2: Als modulare Vermessungsstation bietet der Wild T1600 hohe Leistung und Bedienungskomfort. Das austauschbare REC Modul speichert ca. 500 Datenblöcke.

Beeindruckendes Bedienungskonzept

Überlegenheit ist dem modularen Wild-Konzept auch in bezug auf die Bedienung des Wild T1600 zu attestieren. 14 farbcodierte und eindeutig beschriftete Tasten sind mit den Anzeigefeldern verwechslungsfrei und gut sichtbar an zentraler Stelle angebracht. Steuerung, Stromversorgung, Anzeige, Daten- und Code-Eingaben sowie Registrierung sämtlicher Werte erfolgen durch zentrale Befehle direkt ab T1600. So löst ein einziger Druck der ALL-Taste gleichzeitig Winkel- und Distanzmessung, Berechnung programmierter Funktionen und Registrierung der Werte auf REC Modul oder Datenterminal GRE aus.

Ein Wort noch zur Stromversorgung: die aufladbare Einschubatterie des T1600 ermöglicht ununterbrochene Winkelmessungen während sieben Stunden oder kombiniert mit dem Wild DI1000 bei angeschlossener externer GEB71 Universalbatterie etwa 3500 Winkel- und Distanzmessungen.

Kurz: dieser Wild T1600 ist nicht nur ein überragender elektronischer Theodolit, sondern auch als modulare Station mit seiner Leistungsstärke, Kompaktheit und Universalität bekannten Totalstationen überlegen. Eine zwar grosse, aber laut Hersteller sofort belegbare Behauptung.

Wild + Leitz AG, Forchstrasse 158,
CH-8032 Zürich

Informatik Informatique

Jochen Ludewig:

Sprachen für die Programmierung – eine Übersicht

194 Seiten mit Abbildungen. Bibliographisches Institut, Mannheim 1985, Hochschultaschenbücher Band 622, Fr. 22.20.

Kürzlich habe ich dieses Taschenbuch gelesen, möchte es sehr empfehlen und mit den nachfolgenden Auszügen aus dem Inhalt das Interesse dafür wecken. H. J. Matthias

Aus dem Vorwort

Dieses Buch ist aus einer Vortragsreihe in den «Technischen Abendkursen Baden» entstanden. Das ungewöhnlich grosse Interesse an diesem Thema war Anlass, die zu den einzelnen Vorträgen verteilten Unterlagen zusammenzufassen und zunächst als Bericht, nun schliesslich mit erheblichen Erweiterungen als Buch herauszubringen.

Der Stil ist durch die Vorgeschichte geprägt: kein erschöpfendes Lehrbuch, sondern eine Einführung, keine Hauptmahlzeit, sondern eine Vorspeise, die hoffentlich Appetit macht. Es wendet sich damit an alle, die keine oder nur einzelne Programmiersprachen kennen und sich einen Überblick des Gebiets verschaffen wollen. Besonders Gewicht ist auf die Wechselwirkung zwischen Sprachen und Programmier-techniken gelegt.

Die Auswahl einer Programmiersprache hat Ähnlichkeit mit der Wahl eines Autos: Zwar gibt es viele technische Merkmale, aufgrund derer man vergleichen kann, doch geben letztlich oft irrationale Momente den Ausschlag. Das sich Programmiersprachen nicht bildlich darstellen lassen, habe ich in den sonst allzu trockenen Text eigene (Ab-)Zeichnungen von Fahrzeugen eingestreut, die – nach meinem Empfinden – gewisse Übereinstimmungen mit den behandelten Sprachen haben. (Abb. 1)

Aus dem Inhaltsverzeichnis

1. Einleitung

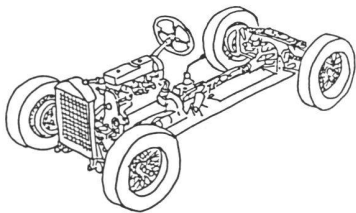
2. Frühe Programmiersprachen
3. Programm-Qualitäten
4. Blockorientierte Sprachen
5. Strukturierte Programmierung und Struktogramme
6. Sprachen mit Modul- und Prozess-Konzept
7. Programm-Dokumentation
8. Sprachen für die Systemprogrammierung
9. Interpretierbare Sprachen
10. Nicht-konventionelle Sprachen
11. Klassifikation und Definition von Programmiersprachen
12. Abschliessende Bemerkungen
13. Literatur

Aus der Einleitung

Niemand weiss, wieviele Programmiersprachen es gibt. Zeitweise gehörte es geradezu zum guten Ruf eines Informatikers, seine eigene Sprache zu erfinden. Auch wenn man von den vielen so entstandenen Exoten absieht, bleiben mehrere Dutzend Sprachen übrig (mit Dialekten einige Hundert), die tatsächlich verwendet werden. Aus diesem Grunde hat Jean Sammet den Turm zu Babel als Symbol für die Situation bei den Programmiersprachen verwendet (Abb. 2). Der Schwerpunkt dieses Buches liegt bei den weit verbreiteten, relativ allgemein verwendbaren Programmiersprachen («general purpose programming languages») für konventionelle Rechner.

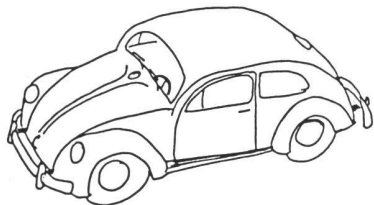
Aus «Kriterien für gute Programmiersprachen»

1. Namen
 - 1 keine Beschränkungen bei der Länge von Namen
2. Konstanten und Typen
 - 1 Definition symbolischer Konstanten und Typen
 - 2 Aufbau komplexer Datentypen
 - 3 strikte Typprüfung (Strong Typing)
 - 4 Unterstützung bei der Bildung abstrakter Datentypen
3. Wertzuweisungen
 - 1 Nebenwirkungen ausgeschlossen oder scharf kontrolliert
4. Ablauf
 - 1 sichere, geschlossene Konstrukte für den Ablauf
 - 2 Prozeduren mit nach Kommunikationsrichtung qualifizierten Parametern
 - 3 keine allgemeine Sprunganweisung (GOTO), stattdessen Aussprung und Ausnahme-Behandlung (Exception Handling)
 - 4 höheres Sprachkonzept für die Prozesskommunikation
5. Stil und Umfang
 - 1 nach einem eingängigen Prinzip aufgebaut (nicht zusammengestoppelt)
 - 2 überschaubar, leicht erlernbar
 - 3 an mathematischem Formalismus orientiert
6. Übersetzer
 - 1 rechtzeitig auf den richtigen Maschinen verfügbar



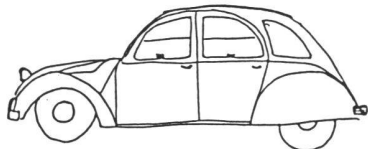
Assemblersprachen / Fahrzeug-Chassis

Die Basis für alles andere, aber selbst nur für Spezialisten und Bastler brauchbar



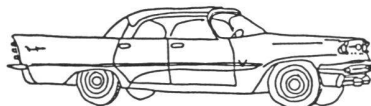
FORTTRAN / VW Käfer

Er läuft und läuft und



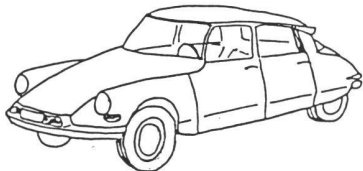
COBOL / Citroen 2CV

Eine spezielle und sehr beehrte Weltanschauung



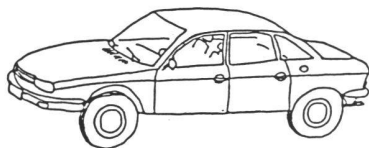
PL/I / De Soto Fireflite Sportsman

Viel Komfort, wenig Sicherheit



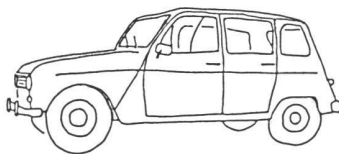
Algol 60 / Citroen DS 19

Moderner als viele seiner Nachfolger



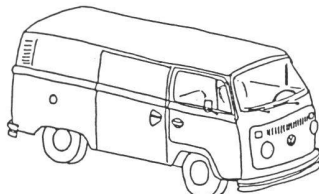
Algol 68 / NSU Ro 80

Erfolgreich nur als Ideen-Lieferant für andere



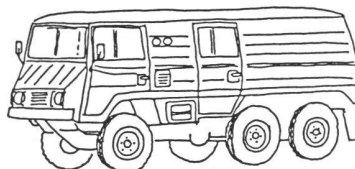
PASCAL / Renault R 4

Einfach, Handlich und zweckmässig



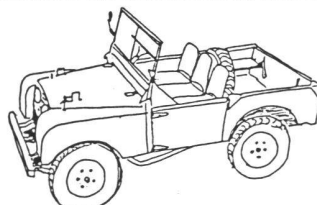
MODULA-2 / VW-Transporter

Spartanisch in der Grundausstattung, aber ausbaufähig und auch grösseren Aufgaben gewachsen



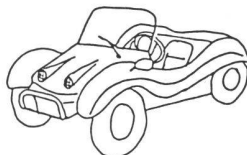
Ada / Steyr-Daimler-Puch Pinzgauer 6x6

Aus dem militärischen Bereich stammend und auf Schwerstarbeit zugeschnitten



C / Land-Rover

Viel Flexibilität speziell für bodennahe Aufgaben



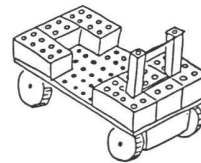
BASIC / VW-Buggy

Einstiegsdroge für ein falsches Verhaltensmuster



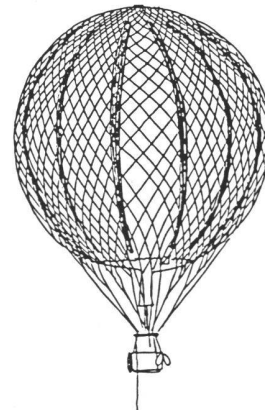
LISP / Bell-Helikopter 47 G

Schwer zu handhaben, aber auf speziellen Gebieten unschlagbar



LOGO / LEGO-Auto

Mehr als ein Spielzeug aus einfachen Bausteinen



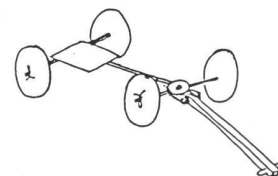
Smalltalk-80 / Freiballon

Die Steuerung ist unsichtbar



PROLOG / fliegender Teppich

Traumhaft, aber noch nicht genügend tragfähig



Turing-Maschine / Handwagen

Auf das absolut Notwendige beschränkt

Abb. 1

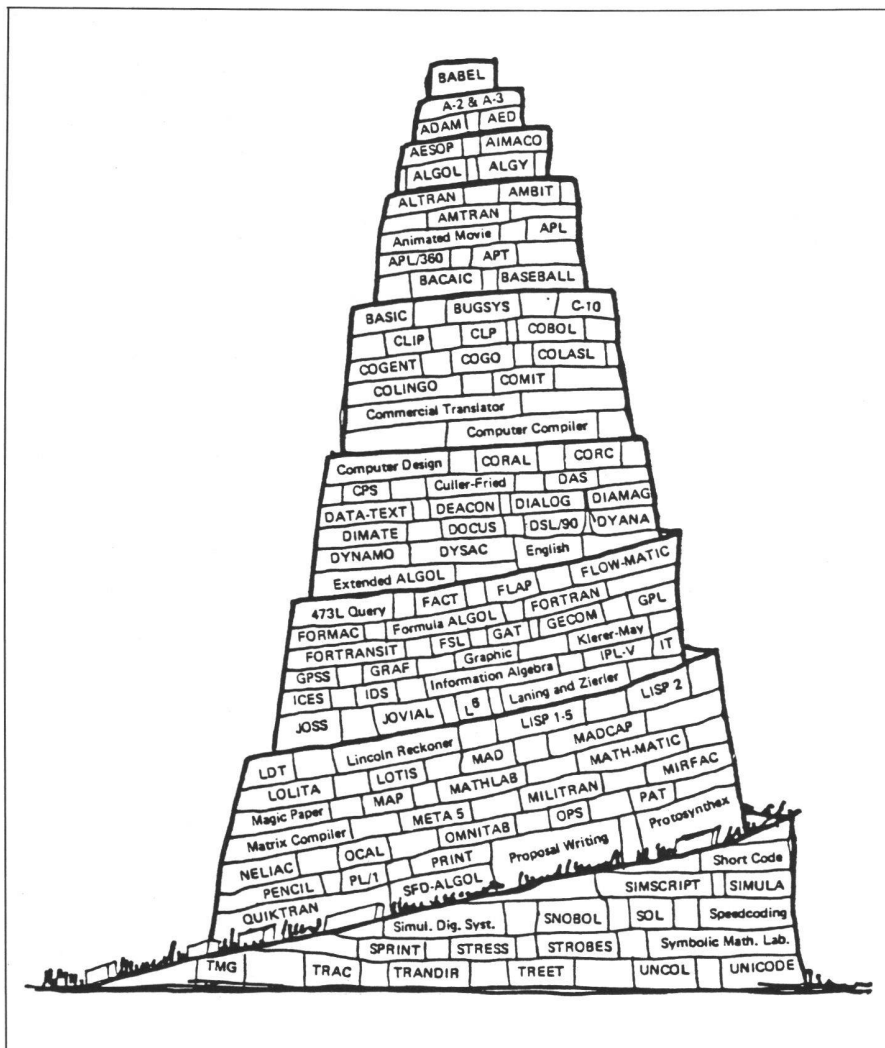


Abb. 2: Der Turm zu Babel aus Programmiersprachen (aus J. Sammet, 1969)

- 2 benutzerfreundliche Fehlerbehandlung
- 3 einheitliche Bedienschnittstelle
- 4 separate Übersetzung
- 5 erzeugt hinreichend effizienten Maschinencode.

Aus «Kriterien für gute Programmierer»

Das wichtigste Kennzeichen eines guten Programmierers ist seine *Disziplin*. Dank ihrer widersteht er dem Drang, so schnell wie möglich zu codieren und zu testen, legt stattdessen seine Programme gründlich und sauber an und dokumentiert vor der Codierung. Unzulänglichkeiten der Programmiersprache kann der Programmierer durch Disziplin teilweise ausgleichen, indem er, wo nötig, «so tut als ob», also etwa in FORTRAN GOTO nur so einsetzt, wie es in MODULA möglich ist usw. Die Disziplin veranlasst ihn auch, sich an Regeln und Normen zu halten und seine Programme nach dem Problem zu formen, nicht nach sich selbst («ego-free-programming»). Schliesslich schützt ihn die Disziplin von Unbeständigkeit und Schwäche gegenüber wechselnden Wünschen aus der Umgebung (er kann NEIN sagen). Ein guter Programmierer hat ein intuitives Verständnis dafür, was *Komplexität* bedeutet. Der Unterschied zwischen 10 Zeilen und 10 000 Zeilen besteht für ihn nicht nur in

einem Faktor Tausend. Er weiss, welche *Qualität* des Problems durch die Komplexität hinzukommt, und er richtet sich auf die damit verbundenen Schwierigkeiten ein.

In vielen Fällen wird der Programmierer von einer soliden Grundlage in Mathematik profitieren (Algebra, Automatentheorie usw.); fehlt dieser Teil der Ausbildung, so lässt er sich neben der Praxis kaum noch nachholen.

Auch für den guten Programmierer ist die wichtigste Sprache die natürliche; sowohl bei der Zusammenarbeit im Team als auch beim Dokumentieren formuliert er *klar, knapp und präzise*.

Um mit einer sich schnell entwickelnden Technik Schritt zu halten, ist ein guter Programmierer *flexibel und lernfreudig*; er ist *kooperativ* und auch bereit, seine Arbeit der *Kritik* seiner Kollegen auszusetzen.

Aus dem Schlusswort

Meine eigene Bewertung hat schon durch viele Zeilen dieses Buches geblinzelt, ich möchte sie auch explizit mitteilen: Für die Grundausbildung kommen vor allem PASCAL und MODULA-2 in Frage. Auch für die Systemprogrammierung ist MODULA-2 sehr gut geeignet, C erscheint mir nur als Alternative zu Assemblersprachen oder BA-

SIC attraktiv. Bei grossen Projekten werden die Vorteile von ADA voll zur Geltung kommen. Im konkreten Fall kommen natürlich nur Sprachen in Frage, für welche geeignete Programmiersysteme (Übersetzer, Binder usw.) vorhanden sind oder beschafft werden können.

In der Zukunft erwarte ich, dass sich objektorientierte Sprachen wie SMALLTALK zu einer ernsthaften Konkurrenz der konventionellen Sprachen entwickeln. PROLOG und seine Nachfolger werden dagegen der Programmierung neue Anwendungsbereiche erschliessen.

Es sollte nach allen Aussagen dieses Buches klar sein, dass Programmieren einerseits keine «schwarze Kunst» ist, andererseits aber auch kein Kinderspiel, das jeder Laie nach ein paar Wochen in einem Programmierkurs – oder schlimmer noch nach einigen Versuchen am Home-Computer – beherrscht. Programmieren ist ein «Kopferwerk», also ein Handwerk des Kopfes, das man solide erlernen und anwenden sollte. Daran ändert auch die Tatsache nichts, dass heute in der Praxis viel Pfuschi üblich ist.

Recht / Droit

Die Unvorhersehbarkeit von enteignenden Immissionen

Nach der bundesgerichtlichen Rechtsprechung, die trotz Kritiken aufrechterhalten geblieben ist, sind störende Einwirkungen des Strassen- oder Eisenbahnverkehrs auf benachbarte Grundstücke selbst dann, wenn sie im Sinne von Art. 684 des Zivilgesetzbuches (ZGB) übermässig sind, nur unter ganz bestimmten Voraussetzungen ein Anlass zu einer Entschädigung wegen materieller Enteignung: Dann nämlich, wenn diese Immissionen für den Grundeigentümer unvorhersehbar sind, wenn sie ihn in besonderer Weise beeinträchtigen und ihm schweren Schaden verursachen. Ob eine solche Immission voraussehbar war oder nicht, gibt indessen immer wieder etwa Anlass zu gerichtlichen Auseinandersetzungen. Beispiele helfen, darüber Klarheit zu gewinnen.

Das Bundesgericht verlangt, dass ein Grundeigentümer vom Moment an, in dem ein immissionsträchtiges Verkehrswegprojekt bekannt ist, diesem Rechnung trägt. Das Bundesgericht hat bereits erklärt, ein Grundeigentümer könne keine Entschädigung erwarten, weil sein Eigentum entwertet werde, wenn er eine Baulandparzelle kauft, während ein solches Projekt bereits bekannt ist. Ebenso wenig kann er die Vergütung des Wertverlusts eines Gebäudes erwarten, das er in Kenntnis der Planung einer nahen Immissionsquelle erstellt. Im Bundesgerichtsentscheid BGE 111 Ib 235, Erwägung 2a, ist ausserdem festgehalten worden, dass ein Eigentümer, der ein Grundstück dank Erbvorzugs erlangt, genau wie ein Erbe in die Rechtsstellung seines Rechtsvorgängers eintritt.