Man kann das Programmieren lernen und dann DV-Maschinen sinnvoll nutzen, ohne etwas über die technischen und logischen Abläufe im Rechner zu wissen. Andererseits läßt sich das Erlernen der Programmierung aber auch als Werkzeug einsetzen, mit dem Funktionsweise und Struktur von elektronischen Rechnern verständlich werden. Der zweite Weg wird hier beschrieben. Hervorzuheben ist die günstigere Beurteilung von Flußdiagrammen gegenüber Struktogrammen. Schneider beschreibt in seinem Aufsatz „Grafische Darstellung von Programmabläufen" im Mikrocomputer-Teil dieses Jahrbuches weitere Erfahrungen dazu.

H. Nahrstedt

Programmierbare Taschenrechner und Mikrocomputer in der Erwachsenenbildung

Erfahrungen an der VHS Hamm

Die Volkshochschulen bemühen sich von Jahr zu Jahr, ihr Angebot weiter auszubauen. So wurde erstmalig im Herbst 77 an der VHS Hamm der Kursus „Einführung in das Programmieren von Taschenrechnern" eingeführt. Im Herbst 78 kam noch ein Kursus „Mikrocomputer" hinzu. Die dabei gemachten Erfahrungen haben unser Konzept ständig verändert. Mit einem Kursus „Programmieren in BASIC" hoffen wir unsere Palette ab Herbst 79 abzurunden.

Wegen der oftmals unterschiedlichen oder falschen Erwartungen, mit denen unsere Teilnehmer zu uns kommen, möchte ich an dieser Stelle die Gelegenheit ergreifen und die Vorstellungen über diese Kurse korrigieren. Vielleicht ist dies aber auch ein Anreiz für manchen Leser, sich auch für anderen Orts bestehende Kurse zu interessieren.

Da ist zunächst der Hörerkreis zu nennen, der in dieser Komplexität und mit den Motivationen wohl in keiner anderen Schulform auftritt. Angefangen von einer Hausfrau, die ihr Budget gern mit einem Taschenrechner überprüfen möchte und deren Küchengeräte bereits per Mikrocomputer gesteuert werden, über den Schüler, der einen programmierbaren Taschenrechner im Unterricht einsetzt und in einer Arbeitsgemeinschaft an einer Tischrechenanlage lernt, bis hin zum Ingenieur, dessen ständige Berechnungen ein programmierbarer Taschenrechner erleichtern kann und dessen geplante Maschinen durch einen Mikrocomputer gesteuert operieren sollen, ist alles vertreten. Dieser Hörerkreis verlangt eine Normierung, d.h. es muß vor dem eigentlichen Thema soviel Grundwissen vermittelt werden, daß alle „die gleiche Sprache sprechen".

Sinn und Ziel dieser Kurse ist es allerdings nicht, ein fundiertes Grundwissen zu vermitteln, sondern vielmehr Einblick in ein neues Wissensgebiet zu geben und einen Grundstein zu legen für eine spätere, gezielte Fortbildung in Schulungskursen oder auf autodidaktischem Wege. Es wird gerade dem Anfänger ermöglicht, die besonders auf dem Computermarkt bestehende Informationsflut zu übersehen und die für ihn interessanten Informationen richtig einzuordnen. Dabei können oftmals falsche Meinungen revidiert und fälschlich angefangene Bildungswege umgelenkt werden.

Den Lerninhalt beider Kurse kann man grob in zwei Hälften teilen. Der 1. Teil befaßt sich mit der Betrachtung allgemeiner *Strukturen und Regeln*, und im

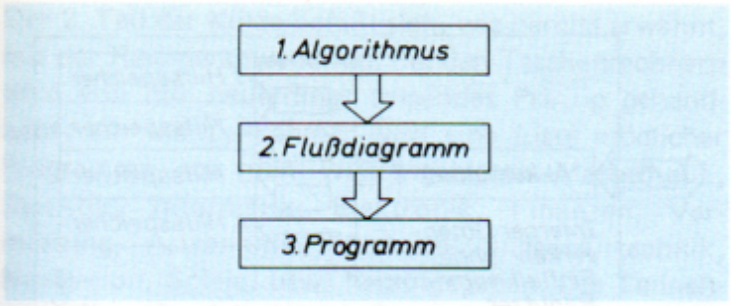

Bild 1 Der Weg von einer Idee zum Programm

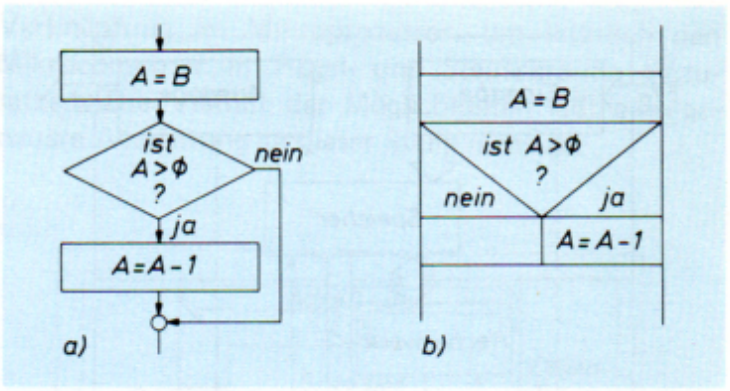

Bild 2 Vergleich Flußdiagramm (a) und Struktogramm an einer bedingten Verzweigung (b)

2. Teil erfolgt dann die Hinwendung zu konkreten *Computertypen*. Während man den 1. Teil wiederum konkret nach *Software* (Programmierung) und *Hardware* (Computerstrukturen) trennen kann, setzt im 2. Teil die Programmierung auch Kenntnisse der vorhandenen **Rechnerstruktur** voraus.

Es wird nachfolgend die Kursusstruktur weiter detailliert. Der Softwareanteil des 1. Kursusteiles beginnt mit der Definition des Begriffs **Algorithmus** und der praktischen Demonstration an Beispielen. Eins der klassischen Beispiele ist der Euklidische Algorithmus zum Finden des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen. Hier werden Grundlagen zur **linearen Programmierung** gelegt, Einblicke in den zeitlichen Ablauf einer Problemlösung gegeben und Begriffsvorstellungen darüber geschaffen, welche Probleme durch Computer lösbar sind.

Die Übersetzung eines Algorithmus ins *Flußdiagramm* ist der zweite Schritt auf dem Weg zum fertigen Programm, wie es **Bild 1** graphisch wiedergibt. Dabei habe ich immer wieder festgestellt, daß das Aufstellen eines Flußdiagramms allen Teilnehmern am leichtesten fällt. Es liegt wohl am graphischen Anteil, denn man hat sozusagen „auf einen Blick" alle Zusammenhänge erfaßt. Die alternativen **Struktogramme** sind hier weniger wirksam. Außerdem sind sie für einfache Programmiersprachen auch weniger geeignet. **Bild 2** verdeutlicht an einem Beispiel beide Methoden. Nach mehreren Übungen wird in der Regel die schriftliche Formulierung eines Algorithmus fallengelassen und die Teilnehmer beginnen gleich mit dem Flußdiagramm. Allgemein wird immer die Übersichtlichkeit bei der Entwicklung im Flußdiagramm hervorgehoben. Der dritte Schritt, die Übersetzung des Flußdiagramms in das jeweilige *Rechnerprogramm* fällt dann am leichtesten, wenn zuvor die Rechnersprachelemente an Beispielen erklärt werden. Hier ist es für die spätere Übersichtlichkeit eines Programms wichtig, daß man gewisse Grundregeln und Gesetzmäßigkeiten einführt. Zum Rüstzeug für das Programmieren gehören weiterhin der Aufbau von **bedingten Sprüngen** und *Unterprogrammen*, sowie das Prinzip der **indirekten und relativen Adressierung**. Eventuell muß man bei Rechnern, die über bestimmte oftmals notwendige Grundfunktionen nicht verfügen, Hilfsprogramme einführen.

Bei der *Hardware* kann man für alle Gebiete von einer Grundstruktur, wie **Bild 3** sie zeigt, ausgehen. Diese wird dann in den einzelnen Richtungen konkretisiert. Bei den programmierbaren Taschenrechnern ist es insbesondere die Betrachtung der *Arbeitsspeichereinheit* (ASE) **Bild 4.**

Rechnerstruktur: Logischer Aufbau eines DV-Systems.

Algorithmus: Rechenvorschrift, festgelegtes Schema für einen Lösungsweg.

Lineare Programmierung: Entwickeln und Schreiben eines Programms in der logischen Folge, die dem Ablauf des zu programmierenden Vorgangs entspricht.

Struktogramm: Grafische Darstellungsmöglichkeit für Programmabläufe, wobei nicht die Reihenfolge im Vordergrund steht (wie beim Flußdiagramm), sondern die Programmstruktur (s. *Schneider*: Grafische Darstellung von Programmabläufen).

Bedingter Sprung: Verzweigung in einem Programmablauf, die durch Abfrage einer vorgegebenen Bedingung zustande kommt.

Indirekte Adressierung: Während „normalerweise" im Adreßteil eines Programmbefehls direkt die Speicherzelle angegeben ist, in der die gewünschten Daten stehen (*direkte Adressierung*), wird hier eine Speicherzelle genannt, die erst die Adresse enthält.

Relative Adressierung: Die Zieladresse, die den nächsten auszuführenden Befehl oder gewünschte Daten enthält, wird hierbei als Abstand, also relativ zu einer festgelegten Adresse angegeben.

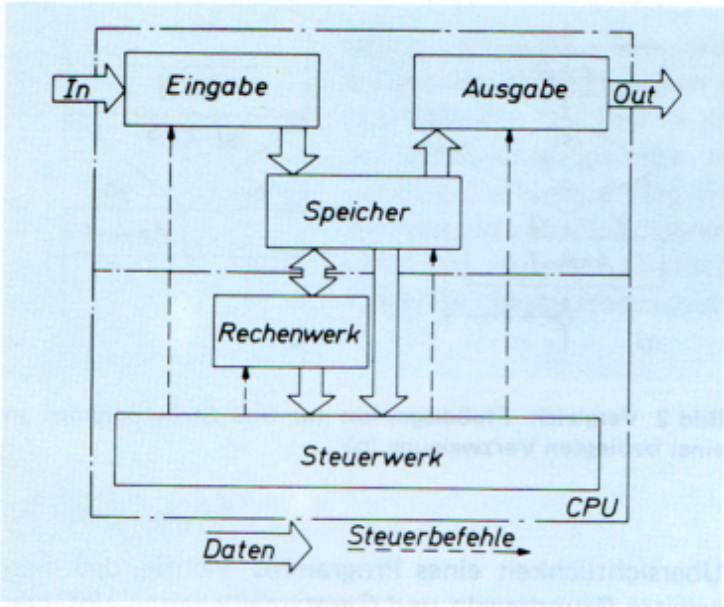

Bild 3 Grundstruktur eines Computers (In: Eingabe; Out: Ausgabe)

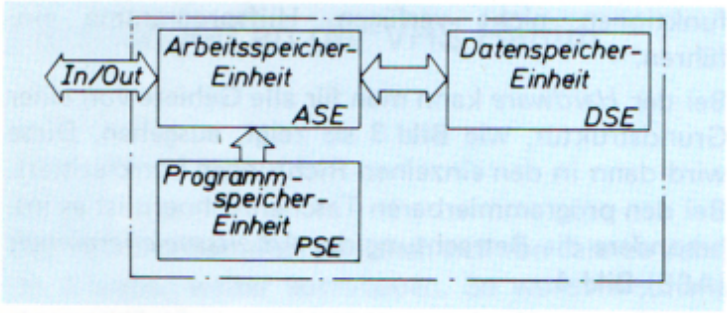

Bild 4 Gliederung der Speichereinheit von programmierten Taschenrechnern

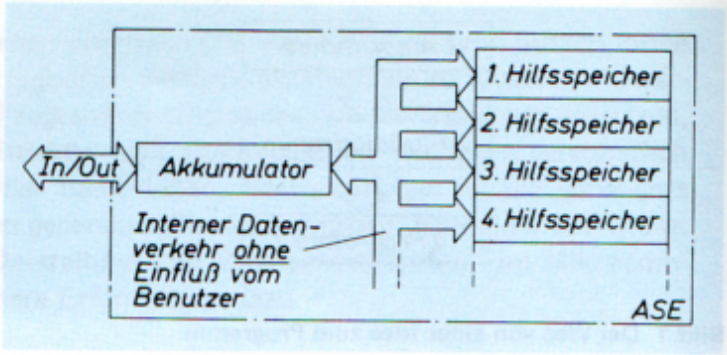

Bild 5 Arbeitsspeichereinheit (ASE) bei algebraischer Notation

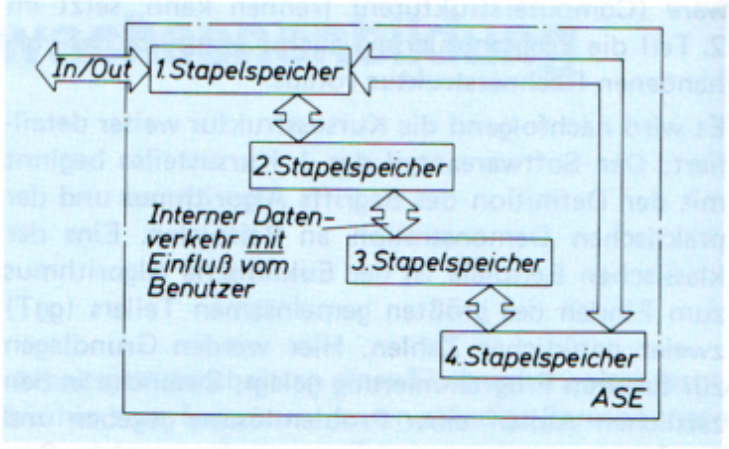

Bild 6 Arbeitsspeichereinheit (ASE) bei umgekehrter polnischer Notation

Je nach der *Notation des programmierbaren Taschenrechners* handelt es sich bei der Arbeitsspeichereinheit (ASE) um einen **Akkumulator** mit zusätzlichen Hilfsregistern bei **algebraischer Notation (Bild 5)** oder um einen Stapelspeicher **(Stack)** bei **umgekehrter polnischer Notation (Bild 6).**

Bei den *Mikrocomputern* kann man zu einem einfachen Blockbild wie **Bild 7** es zeigt übergehen, bei dem man danach bereits sehr konkret auf einen bestimmten *Mikroprozessor* zu sprechen kommen muß. Eine Alternative ist die Einführung eines hypothetischen Mikroprozessors mit minimaler Ausstattung, den man je nach Wissensstand weiter ausbaut, bis seine Struktur dem vorliegenden Mikroprozessor ähnelt. Danach kann man zum Computer wechseln.

CPU: *Central Processing Unit*; Zentraleinheit einer EDV-Anlage.

Akkumulator: Das wichtigste Arbeitsregister in der CPU, in dem die meisten Operationen ausgeführt werden.

Stack: Spezieller Speicherbereich (*Stapelspeicher*), der in der Regel durch die CPU automatisch genutzt wird (z.B. bei Programmunterbrechungen oder Unterprogrammaufrufen).

Algebraische Notation: Von den meisten Taschenrechnern genutztes Verfahren zur Eingabe und Verarbeitung, z.B. a + b =: Eingabe in dieser Reihenfolge.

Umgekehrte Polnische Notation: Eingabemethode, die ohne Klammersetzung auskommt und „Punktrechnung" vor „Strichrechnung" richtig behandelt.

Der 2. Teil der Kurse befaßt sich, wie bereits erwähnt, mit der Rechneranwendung. Bei den Taschenrechnern wird von mir neuerdings folgendes Prinzip gehandhabt: An die Teilnehmer wird eine Liste möglicher Programme aus den Wissensgebieten Mathematik, Statistik, Informatik, Elektronik, Finanzen, Vermessung, Astronomie, Mechanik, Ingenieurtechnik, Navigation, Spiele, usw. herausgegeben. Die Teilnehmer haben dann die Möglichkeit, von ihnen gewünschten Programme anzukreuzen oder eigene Vorschläge zu unterbreiten, die dann mit in eine neue Übersicht aufgenommen werden. Außerdem wird der eigene, vorhandene Taschenrechner angegeben. Die gewünschten Programme werden so später im Kurs auf diesen zugeschnitten, und der Teilnehmer hat damit optimale Anwendungsbeispiele. Darüber hinaus wird den Teilnehmern ein Einblick in die verschiedenen Notationen und Programmiermöglichkeiten der auf dem Markt befindlichen Rechner vermittelt.

Beim Mikrocomputer-Kursus gehen wir von einem einzigen Typ aus, dem Prozessor 6502, der sich wiederum im PET 2001 befindet. Hier ist die besondere Vielfalt von Anwendungsmöglichkeiten wichtig. Da gehört zunächst das Benutzen vorhandener **Peripheriegeräte** zur Einführung. Alsdann betrachten wir die Ein-Ausgabe analoger und digitaler Werte und ihre Verknüpfung im Mikroprozessor, um letztlich den Mikrocomputer in Regel- und Steuerkreisen einzusetzen. Die Vielfalt der Möglichkeiten läßt eine genauere Ausführung an dieser Stelle nicht zu.

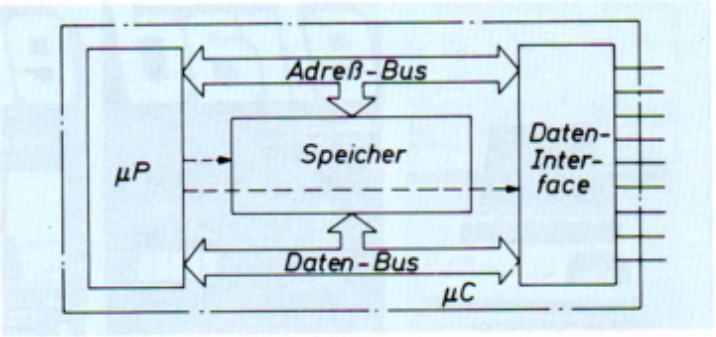

Bild 7 Blockbild eines einfachen Mikrocomputers

Peripherie: Alle Bauteile und Geräte einer EDV-Anlage, die außerhalb der CPU angeordnet sind (z.B. Bedienungsgeräte, externe Speicher).

Interface: Schnittstelle zwischen CPU und Peripherie.

Einteilung von Rechnerklassen

- *Großcomputer* als aufwendigste Spezialrechner für Technik, Wissenschaft und Rechenzentren (z.B. Kerntechnik, Raumfahrt, Zentralregister);
- *Mittlere Anlagen* für den kommerziellen Bereich („Mittlere Datentechnik", MDT);
- *Prozeßrechner* zur Meßdatenverarbeitung in Industrie und Wissenschaft sowie zur Überwachung, Steuerung und Regelung technischer Produktionsprozesse;
- *Minicomputer* für die „Dezentrale Datenverarbeitung" und als Prozeßrechner;
- *Mikrocomputer* für die noch weiter dezentralisierte Datenverarbeitung, Datenvorverarbeitung in „intelligenten Terminals" sowie als anpassungsfähiges Mittel für die Meßtechnik und für einfachere Steuerungs- und Überwachungsaufgaben;
- *Programmierbare Tisch- und Taschenrechner* als bequeme und preiswerte Hilfsmittel in unzähligen Anwendungsfällen.