

Georgische Technische Universität  
Universität Erlangen-Nürnberg

*Prof.Dr. Klaus Myer-Wegener*  
*Prof. Dr. Gia Surguladze*

**GRUNDLAGEN DER PROGRAMMIERUNG**  
(Teil - I)

**unterstützt durch DAAD**  
**(Deutschland)**



**Erlangen–Nürnberg–Tbilissi**  
**2005**

## INHALTSVERZEICHNIS

Einleitung	3
<b>Teil I. Die theoretischen Fragen</b>	<b>5</b>
<b>Lektion 1: STRUKTUR DER PROGRAMMGESTEUERTEN RECHENAUTOMATEN</b>	<b>5</b>
1.1. Kurze historische Übersicht	5
1.2. Struktur moderner Rechenautomaten	7
<b>Lektion 2: ZAHLENDARSTELUNG UND -UMWANDLUNGS- SYSTEME</b>	<b>9</b>
2.1. Zahlendarstellung in elektronischen Maschinen	9
2.2. Zahlenumwandlungssysteme	11
2.3. Datenverarbeitung durch die Binäroperationen	13
<b>Lektion 3: BOOLESCHE ALGEBRA</b>	<b>15</b>
<b>Lektion 4: ALGORITHMEN UND PROGRAMME</b>	<b>19</b>
LITERATURVERZEICHNIS	22

## EINLEITUNG

Dieses Buch hat sich zum Ziel gesetzt, den Studenten/Innen eine Einführung in C zu bieten, die noch keine oder eine geringe Programmiererfahrung haben. Es werden lediglich die grundlegenden Kenntnisse im Umgang mit dem Betriebssystem gefordert.

Die Sprache C wurde ursprünglich dazu entwickelt, Betriebssysteme zu schreiben und richtet sich deshalb an fortgeschrittene Programmierer. Wenn Sie wenig oder keine Programmiererfahrung haben, ist es sehr wahrscheinlich, dass Sie nicht alles auf Anhieb verstehen.

Im Buch werden grundlegende Konzepte der Programmierung vorgestellt. Nach einer kurzen Einführung (I-Teil) in die Begriffswelt «Informatik» und Architektur von Rechenanlagen und Programmen werden am Beispiel der Programmiersprache C die Grundkonzepte (II-Teil) von Programmiersprachen eingeführt:

- Datentypen,
- Operatoren,
- Ausdrücke,
- Ablaufkontrolle (Schleifen und Switch),
- Rekursion,
- Funktionen und
- Zeiger und Felder.

Dann werden komplexe Elemente der Programmierung-C vorgestellt (III-Teil), und zwar:

- Dateien,
- Matrizen (einfache, kramerische, gauss),
- Strukturen,
- Programmstruktur,
- Modularisierung,

Datenorganisations- und Suchverfahren für die sequentiellen (Sequenzzugriff) und direkten Dateien (Direktzugriff).

Im Rahmen der Übungen soll der praktische Einsatz der in der Vorlesung eingeführten Konzepte anhand kleiner Programmierbeispiele geübt werden.

Den Schwerpunkt des C++ Kapitels bilden objektorientierte Analyse und Design. Ihr Ziel ist der Entwurf von Klassen und Objekten mit ihren Merkmalen und Beziehungen.

Hinweise zu graphischen Notationen (in Unified Modelling Language) und Codierungsregeln (mit CASE Werkzeuge) erleichtern die gleichzeitige Erstellung der Dokumentation.

In der traditionellen Software-Entwicklung sind Datenfluß analyse, strukturierte Analyse und funktionale Dekomposition weitgehend akzeptiert. Beim Datenbankentwurf wird Analyse und Design mittels Entity-Relationship-Diagrammen und Normalisierungsverfahren durchgeführt.

Die Schwerpunkte der visualen, objektorientierten Programmierung werden mittels Werkzeuge Borland C++ Builder und Visual Studio .Net, und zwar C#.NET für eigene Entwicklungen in den wichtigsten Bereichen und Technologien betrachtet:

- Windows-Anwendungen (Applicationen),
- Webanwendungen,
- Objektorientierte Programmierung,
- Komponentendesign usw.

# *Teil I: Die theoretische Fragen*

## Lektion 1

### STRUKTUR DER PROGRAMMGESTEUERTEN RECHENAUTOMATEN

#### 1.1. KURZE HISTORISCHE ÜBERSICHT

Eines der ältesten Hilfsmittel für die Zahlenrechnung ist Abakus (antikes Griechenland), bei dem Zahlen mit Hilfe verschiebbarer Kugeln dargestellt werden.

Wesentlich später entstanden Rechenmaschinen, die der Zahlendarstellung in Form einer Folge geschriebener Ziffern angepasst waren. Die ersten derartigen Rechenmaschinen wurden im 17. Jahrhundert von Pascal und von Leibnitz unabhängig gebaut (Französischer Mathematiker und Philosoph Blaise Pascal (1623-1662) erfand 1642 Additionsmaschine, die mit einer Zehnenübertragung ausgestattet war. Deutscher Mathematiker und Philosoph Gottfried Wilhelm von Leibnitz (1646-1716) entwickelte 1673 die erste Rechenmaschine für Multiplikation und Division).

Der englische Mathematiker Charles P. Babbage (1792-1871) entwickelte 1822 eine Differenzenmaschine zur Berechnung von Tabellenwerken. Er faßte bereits den Gedanken einer wesentlich allgemeineren Maschine, die nicht nur ein spezielles Problem, sondern beliebige Probleme nach vorhergehender genauer Planung des Ablaufs ausführen kann. Diese „Programmierung“ ist dann die wesentliche Vorbereitungsarbeit für den „programmgesteuerten Rechenautomaten“, wie eine solche Maschine genannt wird. Babbage nannte sie „analytical engine“.

Weitere Impulse erhielt die Rechentechnik nicht von der Seite des wissenschaftlichen Rechnens, sondern von der Notwendigkeit der statistischen Auswertung größeren Zahlenmaterials und vom kaufmännischen Rechnen her.

Der Deutsch-Amerikaner Dr. Herman Hollerith (1860-1929) ist der Begründer der modernen Lochkartentechnik. Für die Volkszählung in den Vereinigten Staaten 1890 entwickelte er einen Satz Lochkartenmaschinen, der aus einem Karten, einem handbedienten elektromagnetischen Zähler und einer Sortiereinrichtung bestand. Ähnliches gilt für die Buchungsautomaten.

Zwischen 1930 und 1940 wurden unabhängig in Europa und Amerika die Vorläufer der heutigen elektronischen Rechenautomaten, die Relaisrechner, erdacht und gebaut.

In den USA - der Physiker Howard G. Aiken, Professor an der Harvard-Universität entwickelte 1944 einen Sequenzrechner mit dem Namen Harvard Mark 1.

In Deutschland war Professor, Dr. Konrad Zuseder Konstrukteur der ersten funktionsfähigen programmgesteuerten Rechenanlage der Welt.

1943 entstand in den USA der erste elektronische Rechner, bei dem im Inneren der Maschine nur elektrische Impulse zur Informationsübertragung verwendet wurden.

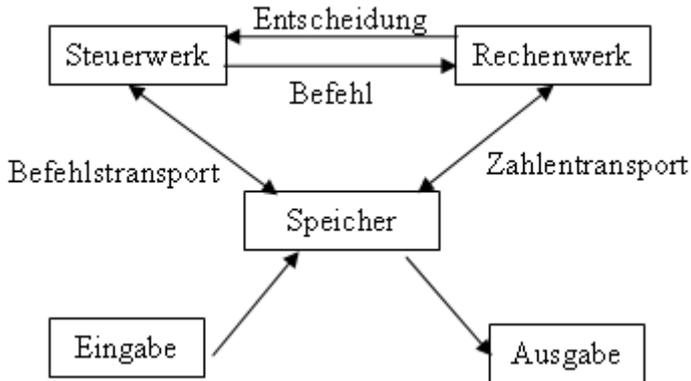
Diese Maschine besteht im wesentlichen aus Rechenwerk und einem Steuerwerk, die mit einem Speicher verbunden sind. In diesem Speicher sind als Impulse sowohl verschlüsselte Zahlen als auch verschlüsselte Rechenanweisungen untergebracht.

Die Gesamtheit der Anweisungen zur Ausführung einer Rechnung wird als Program bezeichnet.

Die weitere Entwicklung verlief rapide, und es entstanden sehr große Rechenautomaten, aber auch Kleinautomaten. Die Programmierung wurde beträchtlich vereinfacht. Es ist hier unter vielen anderen besonders Jon Von Neuman (1903-1957) ungarischer Mathematiker aus den USA zu erwähnen. Heute gibt es eine große Rechenautomatenindustrie, und in der ganzen Welt wird intensiv an der Weiterentwicklung gearbeitet.

## 1.2. STRUKTUR MODERNER RECHENAUTOMATEN

Die Konzeption eines modernen Automaten wurde 1945 erstmalig von J. Von-Neuman angegeben (Abb.1.1).

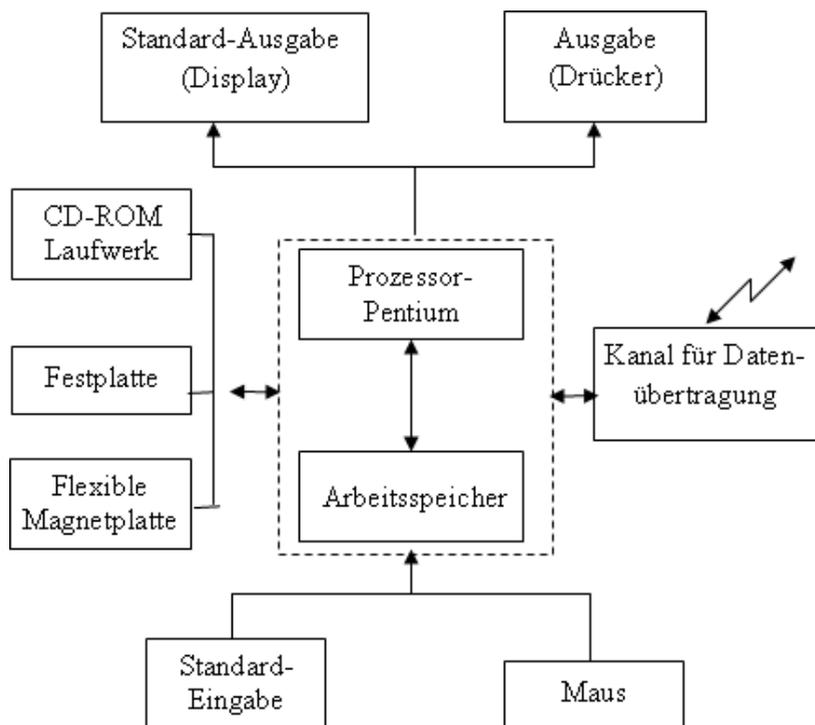


**Abb. 1.1. Grundstruktur eines Rechenautomaten**

Seit 1981 ist konstruiert und breit verwendet die neue Art von Mikrorechenmaschine, die man als persönlichen Computer bezeichnet. Bis 2000 entwickelte sich diese Klasse des persönlichen Computers (PC) ganz stürmisch, aber die innere Struktur und die Arbeitsprinzipien der Maschine sind wenig geändert. In Abb.1.2. ist eine Struktur des PC-Pentiums dargestellt.

Die Eigenschaften des PC-Pentiums für die Betriebssysteme Windows und Linux sind :

- Display: Monitor VGA oder Super VGA;
- Drucker: LaserJet, DescJet;
- Prozessor: Pentium III,IV,V;
- RAM, DDR Speicher: 256 Mb (512, 1024, . . . );
- Festplatte: 40Gb(min). (80, 120, . . .).



**Abb. 1.2. Struktur des Pentium-PC**

## Lektion 2

### 2.1. ZAHLENDARSTELLUNG IN ELEKTRONISCHEN MASCHINEN

Es ist üblich, Zahlen durch Zeichenfolge darzustellen. Zeichen, als auch die Länge der Zeichenfolge, sind endlich. Länge der Zeichenfolge ist dabei die Anzahl der Zeichen, aus denen sie besteht.

Bei der Notierung von Zahlen im Dezimalsystem werden in der üblichen Schreibweise die Zeichen 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; , ; „+“ und „-“ verwendet.

Bei der halblogarithmischen Zahlendarstellung

$$z = m \cdot 10^e$$

tritt noch die Änderung der zu beschreibenden Zeilenhöhe als „Sonderzeichen“ hinzu.

Die Darstellung

$$z = x_m x_{m-1} \dots x_1 x_0, x_{-1} \dots x_{-n} = \sum_{i=-n}^m x_i \cdot 10^i$$

mit  $x=0, \dots, 9$  bezeichnen wir als Normaldarstellung einer Position Zahl in dezimaler Schreibweise.

Ist die Zahl ganz, also  $n=0$ , so wird das Komma weggelassen.

In der Rechenmaschinen verwendet man ein Dualsystem.

Eine Zahl allgemein kann man in der Form

$$z = \sum_{i=-n}^m x_i B^i$$

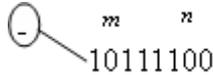
darstellen.

$B=2$  für das Dualsystem (oder Binärsystem).

Die  $x_i$  können die Werte  $\{0, 1\}$  annehmen. Eine Dualstelle ist kürzer **bit** als Abkürzung des Englischen **binary digit**.

Z.B. 00101 ist gleich 5, 10101 ist gleich -5.

Für die negativen Zahlen verwendet man eine zusätzliche Stelle (erste Position) vorsehen, und das Zeichen „+“ durch 0, das Zeichen „-“ durch 1 darstellen. Z.B. die Zahl -3,75 mit  $m=3$  und  $n=4$  wird in der Form



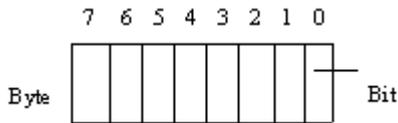
dargestellt.

Hierbei kennzeichnet die erste 1 das negative Vorzeichen, die nachfolgenden Ziffern sind mit Gewichten

$$2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}$$

zu multiplizieren und sämtlich zu addieren.

Eine heute in vielen Datenverarbeitungsanlagen gebräuchliche Einheit für gemeinsam zu verarbeitende Informationen ist das Byte. Es umfasst acht Bits.



Ein Byte ermöglicht die Verschlüsselung von 256 verschiedenen Zeichen (Ziffern, Buchstaben und Sonderzeichen), denn

$$2^8 = 256.$$

Z.B	Character	Binary-Code	ASCII-Code
	a	1 0 0 1 0 1 1 1	97
		9                      7	
	A	0 1 1 0 0 1 0 1	65
		6                      5	
	0	0 1 0 0 1 0 0 0	48
		4                      8	

In den acht Datenbits haben zwei Tetraden (Binärcode für Dezimalziffern) Platz. Sollen mit dem Byte nur Zahlen dargestellt werden, so lassen sich in einem Byte zwei Dezimalstelle verschlüsseln. Diese Verschlüsselungsart bezeichnet man auch als gepacktes Format.

ASCII - Abkürzung für American Standard Code for Information Interchange (Amerikanischer Normkode für Nachrichtenaustausch).

Jedes Tastaturzeichen (Ziffern, Buchstaben, Symbole) hat nur eine einzigen ASCII-Code (Uni-Kode). Damit ist die Vereinbarkeit der übertragende Texte zwischen den verschiedenen Rechenmaschinen unterstützt (z.B. im Internet).

1 Kb (Kilobyte) = 1024 bytes,

1Mb (Megabyte)=1000000 bytes,

1Gb (Gigabyte) = 1000 Mb.

## 2.2. ZAHLENUMWANDLUNGSSYSTEME

Um eine Dezimalzahl in eine Dualzahl umzuwandeln, verwendet man Division (für den ganzen Teil) und Multiplikation mit Basis 2.

Z.B.

$$25,65_{10} \rightarrow (x)_2$$

(:) ← division			⊗ ← Multiplikation		
25	2	1	0,65	2	1 30
12	2	0	0,30	2	0 60
6	2	0	0,60	2	1 20
3	2	1	0,20	2	0 40
1	2	1	0,40	2	0 80
			0,80	2	1 60
			0,60	2	1 20
Rest von unten <u>11001</u>			Rest von oben <u>1010011</u>		

Resultat:  $(25,65)_{10} = (11001,1010011)_2$

Eine Dualzahl in die Dezimalzahl umwandeln:

z.B.  $(100010, 1010111)_2 \rightarrow (x)_{10}$

5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	←die Position
1	0	0	0	1	0	1	0	1	0	1	1	1	

$$2^1 + 2^5, 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-7} = (34,64)_{10}.$$

Ein Zahlensystem mit der Basis 8 (B=8) wird als Oktalsystem, und mit der Basis 16 (B=16) als Hexadezimalsystem (oder Sedezimalsystem) bezeichnet. Für die Datenverarbeitung mit den Rechenmaschinen haben sie große Bedeutung.

Basistabelle Tab.1

Dezimal	Dual	Oktal	Hexa
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
...	...	...	

Alle Umwandlungen werden in der Regel von der Maschine selbst mit Hilfe des speziellen Programms ausgeführt.

Eine Dualzahl in die Hexadezimal umwandeln: z.B.

$$(100010,1010111)_2 \rightarrow (x)_{16}$$

Aus Tab.1:

0010, 0010, 1010, 1110  
 2      2      A      E

Also:

$$x = (22, AE)_{16} = (34,68)_{10} = (42,534)_8$$

### 2.3. DATENVERARBEITUNG DURCH DIE BINÄROPERATIONEN

Alle arithmetischen Operationen einer Datenverarbeitungsanlage, auch Subtraktion, Multiplikation, Division und Wurzelziehen, haben die Addition zur Grundlage.

Additionen werden in der Zentraleinheit ausgeführt. Die Subtraktion wird gewöhnlich auf eine Addition des komplementären Subtrahenden zurückgeführt.

In der Dualsystem haben wir die Operande 0,1.

0+0=0	0-0=0
0+1=1	1-0=1
1+0=1	1-1=0
1+1=10	10-1=1

Z.B.:	0101 (5)	1101 (13) <sub>10</sub>
	+ 0110 (6)	- 0111 (7) <sub>10</sub>
	1011 (11) <sub>10</sub>	0110 (6) <sub>10</sub>

Zur Darstellung der negativen Werte verwendet man das Komplement dieser Zahl.

Z.B. in Dezimalsystem:  $(0)_{10} - (1)_{10} = (-1)_{10}$ ;

In Dualsystem:	_ 0 0 0 0 0 0 0 0	(0) <sub>10</sub>
	0 0 0 0 0 0 0 1	(1) <sub>10</sub>
für Zeichen →	1 1 1 1 1 1 1 1	(-127) → ?

(-127) - es ist ein technischer Fehler bei Subtraktion mit negativen Zahlen.

Das Komplement dieser Zahl  $(-1)_{10}$  im Dualsystem entspricht  $(11111111)_2$ ; Also  $(-2)_{10}$  wird  $(11111110)_2$  sein u.s. w. und  $(-127)_{10}$  wird  $(10000000)_2$  sein.

Jetzt können wir die richtigen Resultate bekommen. Z.B.:

$$\begin{array}{r}
 \phantom{(1111100)} -6 \qquad -4 \qquad -2 \\
 (1111100) - (1111100) = (11111110); \\
 \text{oder} \\
 \begin{array}{r}
 1111111 \quad (-1)_{10} \\
 + 1111110 \quad (-2)_{10} \\
 \hline
 1111101 \quad (-3)_{10}
 \end{array}
 \end{array}$$

In der Regel verwendet man bei Dualzahlen das Zweier – Komplement.

Z.B.: Dezimal: 5-3=2

$$\begin{array}{r}
 \text{Dual normal} \quad 0101 \\
 \phantom{\text{Dual normal}} - 0011 \\
 \hline
 0010
 \end{array}$$

Dual mit Komplement:

$$\begin{array}{r}
 0101 \\
 + 1101 \quad \leftarrow \text{das ist Zweier} \\
 (1) 0010 \quad \text{Komplement von 0011} \\
 \uparrow
 \end{array}$$

**Der Überlauf  
in der höchste Stelle  
bleibt unberücksichtigt**

Bei arithmetischen Operationen kann die Stellenanzahl des Ergebnisses über die Stellenanzahl des Registers, das dieses Ergebnis aufnimmt, hinausgehen. Die nicht mehr unterzubringenden Ergebnisstellen werden als Überlauf bezeichnet.

Beim Programmablauf werden die Prozesse automatisch halten und eine Fehlerreaktion (Anlagenstop oder Starten eines speziellen Programmes - eines Überlaufprogrammes) veranlassen.

## Lektion 3

### BOOLESCHE ALGEBRA

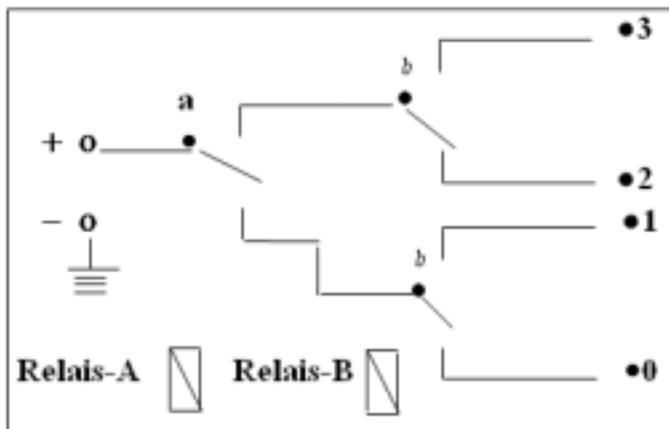
Der englische Mathematiker George Boole (1815-1864) entwickelte eine Algebra, die auch als Algebra der Logik oder symbolische Algebra bezeichnet wird und ursprünglich dafür gedacht war, philosophische Probleme in einer nur zwei Werte (zwei Aussagen) umfassenden mathematischen Formulierung darzustellen.

Den Aussagen wahr (true) und unwahr (false) ordnete er die Zeichen 0 und 1 zu. Auf diesen beiden Aussagen oder Zuständen baute er sein System auf, das es erlaubt, Zusammenhänge zwischen diesen Aussagen bzw. Zuständen in Formeln zu fassen.

Die Gedankengänge der Booleschen Algebra sind später in die Schaltalgebra eingegangen, die der amerikanische Ingenieur und Mathematiker Clod Schannon – einer der wichtigsten Wegbereiter der elektronischen Datenverarbeitung – in den dreißiger Jahren entwickelte. Damit ist die Boolesche Algebra noch heute eine der wichtigsten theoretischen Grundlagen der digitalen Nachrichtenverarbeitung oder Datenverarbeitung.

Die ersten arbeitenden Rechenautomaten mit Hilfsmitteln der Telefontechnik, hauptsächlich Relais bestehen aus einer Spule mit Eisenkern, die über einen Anker Kontakte öffnet oder schließt, je nachdem, ob sie stromlos oder stromführend ist. Mit einem Relais lassen sich also zwei Zustände unterscheiden und damit auch beliebig viele diskrete Zustände, wenn nur genügend viele Relais in passender Weise zusammengeschaltet werden.

Die Kontakte **a,b** sind Wechselkontakte (Abb.1.3). Ist z.B. Relais **B** stromführend, so sind die beiden zugehörigen Kontakte mit Ausgang 1 und 3 verbunden (positiver Pol für **B**), andernfalls mit 0 und 2.



**Abb.1.3. Relaischaltung mit vier Ausgängen**

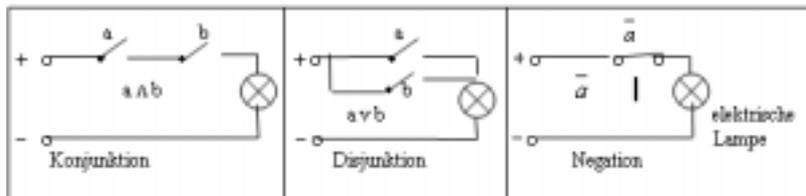
In ähnlicher Weise kann man Relaisschaltungen mit einem Eingang und zehn Ausgängen aufbauen und somit den Ziffern 0 bis 9 gewisse Relaisstellungen zuordnen.

Hieraus lässt sich dann eine Schaltung aufbauen, die aus Relaisstellungen, die den Ziffern zweier Zahlen zugeordnet sind, eine neue Relaisstellung erzeugt, die z.B. der Summe der beiden Zahlen zugeordnet ist.

Auch Speicher können mit Relais aufgebaut werden.

Es kann eine 1 durch einen Kontakt in Arbeitsstellung, 0 durch einen Kontakt in Ruhestellung dargestellt werden.

Die Schaltalgebra (Boolesche Algebra) operiert mit den zwei Grundverknüpfungen: Konjunktion und Disjunktion, denen z.B. in Kontaktschaltungen die Reihenschaltung und die Parallelschaltung von Kontakten zugeordnet sind (Abb.1.4).



**Abb.1.4. Beispiele für Kontaktschaltungen Ausgängen**

Eine wesentliche Rolle spielt die Negation, die als Funktion ausgedrückt werden kann, die 0 auf 1 und 1 auf 0 abbildet. Im Relaischaltungen wird die Negation durch ein Relais mit einem Ruhekontakt realisiert. Ordnet man dem stromdurchflossenen Relais eine 1 zu, dem stromlosen Relais eine 0, dem geschlossenen Kontakt eine 1 und dem offenen Kontakt eine 0, so wird gerade die gewünschte Abbildung erzeugt.

Basis-Operationen für die Boolesche Algebra:

Mit „ $\wedge$ “ bezeichnet man Konjunktion (logische „und“).

Mit „ $\vee$ “ bezeichnet man Disjunktion (logische „oder“).

Mit „ $\neg$ “ bezeichnet man Negation (logische „nicht“).

In der folgenden Tabelle sind die erwähnten Funktionen dargestellt. Die Negation ist eine Funktion einer zweiwertigen Variablen  $x$ ; Konjunktion und Disjunktion sind Funktionen zweier Variablen  $x$  und  $y$ .

Argumente		$\overline{X}$	$\overline{Y}$	$X \wedge Y$	$X \vee Y$	$X \rightarrow Y$
X	Y					
0	0	1	1	0	0	1
0	1	1	0	0	1	1
1	0	0	1	0	1	0
1	1	0	0	1	1	1

$X \rightarrow Y$  ist eine Implikation (if ..., else). „ $\rightarrow$ “ ist gleich  $X \vee \overline{Y}$ .

Z.B. logische Operationen für Dualsystem:

$\neg$	<b>100101</b> ----- 011010	$\wedge$	<b>010111</b> 100101 ----- 000101
$\vee$	<b>010111</b> 100101 ----- 110111	$\rightarrow$	<b>010111</b> 100101 ----- 101101

Für die logischen Umwandlungen verwendet man folgende Regeln (Eigenschaften, Gesetze):

- Reflexivität:

$$X=X \text{ („=“ ist äquivalent).}$$

- Symmetrieeigenschaft:

$$\text{if } X=Y \text{ , then } Y=X.$$

- Transitivität:

$$\text{if } X=Y \text{ and } Y=Z \text{ , then } X=Z.$$

- Idempotente Regeln:

$$X \wedge X=X; \quad X \vee X=X.$$

- Kommutativitätsregeln:

$$X \wedge Y=Y \wedge X; \quad X \vee Y=Y \vee X .$$

- Assoziativgesetz:

$$(X \wedge Y) \wedge Z=X \wedge (Y \wedge Z);$$

$$(X \vee Y) \vee Z=X \vee (Y \vee Z).$$

- Distributivgesetz:

$$X \wedge (Y \vee Z)=(X \wedge Y) \vee (X \wedge Z);$$

$$X \vee (Y \wedge Z)=(X \vee Y) \wedge (X \vee Y).$$

- Negationsgesetz:

$$X \wedge X=0; \quad X \vee X=1.$$

- Doppelte Negation:

$$X=X.$$

- Dualitätsgesetz (Regeln von de Morgan):

$$X \wedge Y= X \vee Y; \quad X \vee Y=X \wedge Y.$$

- Nullelemente:  $1 \vee X = 1; 0 \wedge X = 0.$

- Einzelemente:  $0 \vee X = X; 1 \wedge X = X.$

Manchmal verwendet man:

$$\text{„+“ statt „\vee“, und „\cdot“ statt „\wedge“ .}$$

Z.B.

$$X+Y=Y+X;$$

$$X \cdot Y=Y \cdot X;$$

$$X \cdot (Y+Z)=X \cdot Y+X \cdot Z \text{ und s.w.}$$

## Lektion 4

### ALGORITHMEN UND PROGRAMME

Die Gesamtheit der Regeln, durch deren schematische Befolgung man eine bestimmte Aufgabe lösen kann, wird als Algorithmus bezeichnet. Die Programme, mit denen eine Datenverarbeitungsanlage (DVA) ihre Aufgaben durchführt, sind im Grunde nicht als mehr oder weniger umfangreiche Algorithmen.

Das Programmieren ist dementsprechend das Entwickeln des für die Lösung einer bestimmten Aufgabe mit einer Datenverarbeitungsanlage geeigneten Algorithmus.

Auch im internen Funktionsablauf der Datenverarbeitungsanlage werden Algorithmen, wie z.B. ein bestimmtes Rechenschema für die Multiplikation, verfolgt. Algorithmen sind z.B. auch die Verfahrensvorschriften zum Sortieren oder Mischen von Daten, die im Speicher einer Datenverarbeitungsanlage enthalten sind.

Für die Formulierung von Algorithmen zur Lösung technisch-wissenschaftlicher Probleme mit DVA wurde mit ALGOL (algorithmic language) eine eigene Programmiersprache entwickelt (1950÷1960 Jahren). Abb.1.5 stellt Entwicklungsprozeß verschiedener Hauptlinien von problemorientierten (1), objektorientierten (2), datenbasenorientierten (3) und betriebssystemorientierten (4) Sprachen heraus.

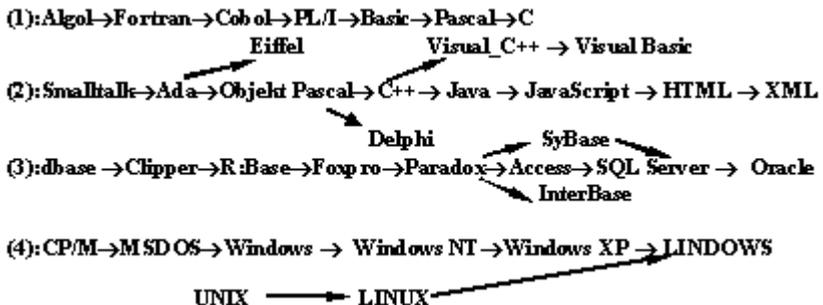


Abb.1.5. Entwicklung der Maschinensprachen und Betriebssystemen

## Das Programm

In der Datentechnik bezeichnet man eine Anweisung oder eine Folge von Anweisungen zur Lösung einer bestimmten Aufgabe als Programm. Dieses Programm steuert die Arbeit einer oder mehrerer zusammenhängender Funktionseinheiten.

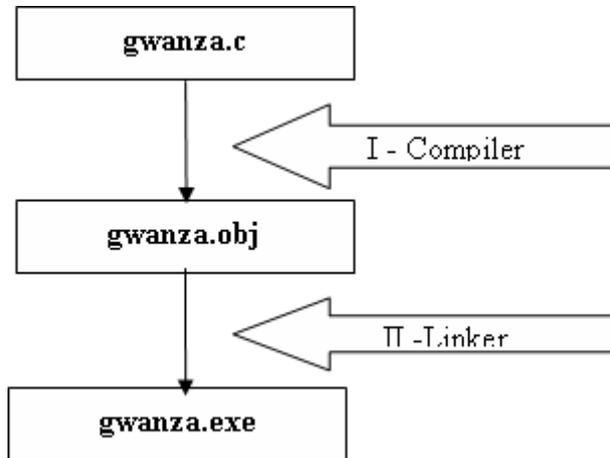
Das Programm setzt sich aus Befehlen zusammen und befindet sich beim Ablauf im Arbeitsspeicher der Anlage. Zur Lösung einer Aufgabe werden von der Zentraleinheit der DVA die Befehle des Programmes nacheinander bzw. nach Maßgabe der von Sprungbefehlen veranlassten Verzweigungen ausgeführt.

Man unterscheidet bei DVA gestreckte Programme und zyklische Programme (Programmschleife). Ein Hilfsmittel zur Verringerung des Programmieraufwandes und des Bedarfes an Speicherplatz für das Programm ist das Unterprogramm, das in einem Hauptprogramm mehrmals verwendet werden kann.

Programme werden zum Teil von den Herstellern von DVA als Software fertig zur Verfügung gestellt (z.B. Betriebssystem) oder vom Anwender der DVA als Anwenderprogramme selbst ausgearbeitet.

Das Ausarbeiten von Programmen wird als programmieren bezeichnet. Ein Programm in der Maschinensprache nennt man Maschinenprogramm (z.B. mit C oder C++). Das in dieser Sprache ausgearbeitete Programm (Quelltext z.B. gwanza.c) wird dann durch ein Übersetzungsprogramm (compiler) in die Maschinensprache übertragen (z.B. gwanza.obj). Das Übersetzungsprogramm gelieferte Maschinenprogramm muß als Programmmodul zuerst mit Hilfe des Binders (linker) in eine ablauffähige Form gebracht werden (z.B. Gwanza.exe).

Z.B., ein C-Quellprogramm besteht aus einer Folge von Funktionsdefinitionen. Jede Funktionsdefinition besteht aus: Ergebnistyp, Funktionsnamen, Parameterliste und Rumpf (s. Abb.1.7).



**Abb.1.6. Phasen des Programmierungsprozesses**

```

/* ----- Kommentaren ----- */
/* Kopf */
| #include < stdio.h >
| #include < ... >
| #define K 100
| #define ...

Ergebnistyp Funktionsname ()
{ /* Rumpf */
  | Parameterliste
  | ...
  | Operatoren
  | ...
}
  
```

**Abb.1.7. Quellprogrammstruktur**

## Literatur

1. Kernighan B. W. , Ritchie D. M.: Programmieren in C, 2. Ausgabe, Carl Hanser, München, Wien; Prentice Hall, London, 1990.
2. K. Meyer-Wegener. Datenverwaltung mit C. Lehrmaterialien. Univ. Erlangen-Nuernberg. 1991.
3. Gogitschaischvili G., Surguladze G., Schonja O. Programmierverfahren mit C&C++. GTU, Tbilisi, 1997.
4. Bothe K., Surguladze G. Moderne Platforms and Languages of Programming (Windows, Linux, C++, Java). GTU, Tbilissi, 2003.
5. Deitel V., Deitel K. Languages of Programming C and C++. Moscow. 2002.