

### 3 Darstellungsformen von Programmen

Viele Problemstellungen sind umfangreich und kompliziert. Systematische Vorarbeit ist nötig, um den Lösungsweg zu beschreiben. Die strukturierte Programmierung verlangt eine gut durchdachte Vorbereitung, bevor mit dem Programmieren begonnen wird.

#### 3.1 Pseudocode

Der Pseudocode ist eine umgangssprachliche Beschreibung des Programmablaufes, die einer Programmiersprache ähnelt. Ein Pseudocode ist nicht genormt, man kann ihn frei formulieren.

Beispiel:

*START*

*LESE  $p, q$*

*BERECHNE  $D = \frac{p^2}{4} - q$*

*WENN  $D < 0$*

*SCHREIBE "keine Nullstelle"*

*SONST BERECHNE  $x_1 = -\frac{p}{2} + \sqrt{D}$*

*BERECHNE  $x_2 = -\frac{p}{2} - \sqrt{D}$*

*SCHREIBE  $x_1, x_2$*

*ENDE*

#### 3.2 Entscheidungstabellen

Entscheidungstabellen sind eine bewährte Technik zur Darstellung komplexer Entscheidungslogiken in Anforderungen. Seit 1979 ist die Entscheidungstechnik in der DIN-Norm 66241 standardisiert.

##### 3.2.1 Aufbau einer Entscheidungstabelle

Eine Entscheidungstabelle besteht grundsätzlich aus vier Elementen oder Quadranten:

- **Bedingungen** beschreiben mögliche Zustände von Objekten.
- **Regeln** oder **Bedingungsanzeiger** zeigen Kombinationen von Bedingungswerten an.
- **Aktionen** geben an, welche Aktivität abhängig von den gegebenen Regeln auszuführen ist.
- **Aktionszeiger** zeigen die Belegung der Bedingungen mit Aktionen an.

Name der Entscheidungstabelle	Regelnummern
Bedingungen	Bedingungsanzeiger
Aktion	Aktionsanzeiger

Zur Erstellung einer Entscheidungstabelle geht man wie folgt vor:

- Aktion angeben
- Bedingung festlegen
- Regeln und Aktionszeiger setzen
- Konsolidierung der Entscheidungstabelle
- Prüfung auf Widerspruchsfreiheit und Vollständigkeit

Eine **vollständige Entscheidungstabelle** liegt vor wenn alle möglichen Bedingungskombinationen im Quadranten "Bedingungsanzeiger" eingetragen sind. Bei **n** Bedingungen gibt es **2<sup>n</sup>** mögliche Bedingungskombinationen.

Ist eine Bedingung erfüllt, dann lautet der Bedingungsanzeiger J (für Ja) bzw. Y (für Yes). Ist sie nicht erfüllt, dann wird N (für Nein oder No) eingetragen.

Im letzten Schritt wird jede Bedingungskombination betrachtet und entsprechend der Problembeschreibung im Quadranten "Aktionsanzeiger" ein Kreuz (X) eingetragen, wenn eine entsprechende Aktion auszuführen ist.

Name der Tabelle	R1	R2	R3	R4	R5	R6	R7	R8
Bedingung 1	J	J	J	J	N	N	N	N
Bedingung 2	J	J	N	N	J	J	N	N
Bedingung 3	J	N	J	N	J	N	J	N
Aktion 1	X				X		X	
Aktion 2		X				X		
Aktion 3	X		X	X		X		X
Aktion 4			X	X			X	
Aktion 5								X

Bei einer Entscheidungstabelle unterscheidet man zwischen zwei Arten der Vollständigkeit. Dabei bezeichnet man eine Entscheidungstabelle, die im Bedingungsanzeigerteil alle möglichen Kombinationen angibt als **formal vollständige Entscheidungstabelle**.

Eine Entscheidungstabelle, die alle praktisch möglichen Kombinationen aufführt wird als **inhaltlich vollständig** bezeichnet.

## Beispiel zweier Entscheidungstabellen

### Formal vollständige Entscheidungstabelle

Scheckeinlösung	R1	R2	R3	R4	R5	R6	R7	R8
Überschreibungsbetrag < 300€	J	N	J	N	J	N	J	N
Zahlungsverhalten einwandfrei?	J	J	N	N	J	J	N	N
Kreditgrenze überschritten?	J	J	J	J	N	N	N	N
Scheck einlösen	X	X			X		X	
Scheck nicht einlösen			X	X				
neue Konditionen vorlegen		X						
unlogisch						X		X

### Inhaltlich vollständige Entscheidungstabelle

Scheckeinlösung	R1	R2	R3	R4	R5	R7
Überschreibungsbetrag < 300€	J	N	J	N	J	J
Zahlungsverhalten einwandfrei?	J	J	N	N	J	N
Kreditgrenze überschritten?	J	J	J	J	N	N
Scheck einlösen	X	X			X	X
Scheck nicht einlösen			X	X		
neue Konditionen vorlegen		X				
unlogisch						

### 3.2.2 Konsolidierung einer Entscheidungstabelle

Vollständige Entscheidungstabellen kann man versuchen zu optimieren. Eine Verdichtung überführt eine vollständige Entscheidungstabelle in eine **konsolidierte Entscheidungstabelle**.

Eine Konsolidierung führt man in folgenden Schritten durch:

1. Existieren Regeln mit identischen Aktionen?

**J** Betrachte zwei dieser Regeln paarweise.

Unterscheiden sich die Bedingungsanzeiger nur in einer Zeile?

**J** Fasse diese beiden Regeln zusammen und ersetzt den unterschiedlichen Bedingungsanzeiger durch den Irrelevanzanzeiger ("–")

**N** Behalte beide Regeln bei.

**N** Die Entscheidungstabelle lässt sich nicht weiter konsolidieren.

Für das obige Beispiel ergibt sich nach der Konsolidierung folgende Tabelle:

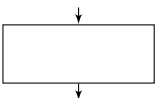
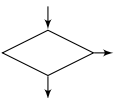
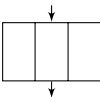
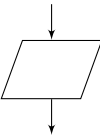
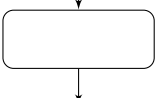
<b>Scheckeinlösung</b>	<b>R1</b>	<b>R2</b>	<b>R3/4</b>	<b>R5/7</b>
Überschreibungsbetrag < 300€	J	N	-	-
Zahlungsverhalten einwandfrei?	J	J	N	-
Kreditgrenze überschritten?	J	J	J	N
Scheck einlösen	X	X		X
Scheck nicht einlösen			X	
neue Konditionen vorlegen		X		
unlogisch				

### 3.3 Programmablaufplan

Der Programmablaufplan stellt für die unterschiedlichen Verarbeitungen **Sinnbilder** zur Verfügung, die durch **Ablauflinien** miteinander verbunden sind.

Sinnbilder gliedern sich in folgende Typen:

- Sinnbilder für Aktionen (Operation, Verarbeitung)
- Sinnbilder für die Ein- und Ausgabe
- das Sinnbild „Ablauflinie“

Sinnbild	Benennung und Bemerkung
	<b>Operation allgemein</b> (process); Insbesondere für die Operation, für die kein besonderes Sinnbild zur Verfügung steht.
	<b>Verzweigung</b> (decision)
	<b>Unterablauf</b> (predefined process); Zusammenfassende Darstellung eines an anderer Stelle definierten Ablaufs, z.B. der Ausführung eines Unterprogramms.
	<b>Eingabe, Ausgabe</b> (input/output); Ob es sich um eine maschinelle oder manuelle Eingabe oder Ausgabe handelt, soll aus der Beschriftung des Sinnbildes hervorgehen.
	<b>Grenzstelle</b> (terminal, interrupt); Innenbeschriftung z.B. „Beginn“, „Ende“.

#### Regeln für das Erstellen von Programmablaufplänen

- Kein Kreuzen der Ablauflinien
- Eingang eines Sinnbildes immer oben

- Ausgang eines Sinnbildes immer unten
- In allen Ablaufplänen, die zu einer Dokumentation gehören, Verzweigungen immer gleich darstellen, d.h. entweder JA- oder NEIN-Zweig immer nach unten führen.
- Bei Schleifen mit Verzweigungen Abauflinie auch nach oben möglich.

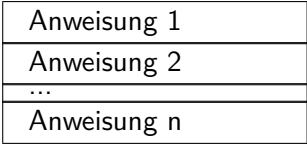
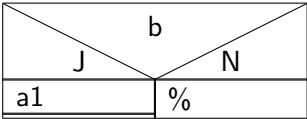
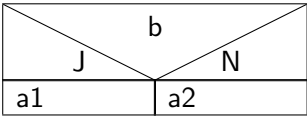
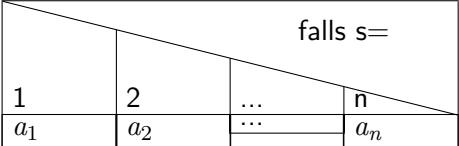
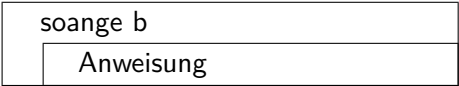
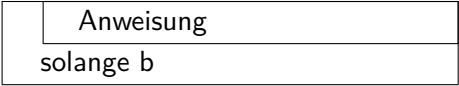
### 3.4 Struktogramme

Ein Nassi-Shneiderman-Diagramm, auch Struktogramm genannt, ist eine Entwurfsmethode für die strukturierte Programmierung, die 1973 von Dr. I. Nass und Dr. B. Shneiderman entwickelt wurde. Es ist genormt nach DIN 66261.

Die grafische Darstellung eines Programmablaufs erfolgt in Form eines geschlossenen Blockes, der entsprechend der logischen Grundstrukturen in untergeordnete Blöcke aufteilbar ist. Ein Struktogramm ist zusammengesetzt aus verschiedenen Symbolen für unterschiedliche Operationsarten.

#### Regeln für die Arbeit mit Struktogrammen

- Die Größe eines Struktogramms ist immer auf eine Seite (DIN-A4) beschränkt.
- In Struktogrammen gibt es immer nur einen Eingang und einen Ausgang. Dies gilt auch für die einzelnen Grundsymbole.
- Der Programmablauf erfolgt grundsätzlich von oben nach unten.
- Die Grundsymbole können ineinander geschachtelt und aneinander gereiht werden.
- Alle Programmverzweigungen laufen an einer Stelle wieder zusammen.
- Bei der Entwicklung von Struktogrammen ist das Prinzip der schrittweisen Verfeinerung anzuwenden (TOP-DOWN).
- Bei größeren Programmen sollte man zusätzlich einen Strukturbaum (hierarchisches Funktionsdiagramm) zur Beschreibung des Gesamtprogramms aufstellen.

Name	Darstellungsform	
	Pseudocode	Struktogramm
Folge (Verbundanweisung)	Anweisung 1 Anweisung 2 ... Anweisung n	
einseitige Auswahl	WENN Bedingung DANN Anweisung	
zweiseitige Auswahl (Alternative)	WENN Bedingung DANN Anweisung 1 SONST Anweisung 2	
mehrseitige Auswahl (Fallunterscheidung)	FALLS Selektor = 1: Anweisung 1 2: Anweisung 2 ... n: Anweisung n	
Wiederholung mit vorange- stelltem Test (mit Eingangsbedingung)	SOLANGE b, FÜHRE Anweisungen AUS	
Wiederholung mit nachge- stelltem Test	WIEDERHOLE Anweisungen SOLANGE b	
gezählte Wiederholung (Zählschleife)	FÜR i = anfw BIS endw mit SCHRITTWEITE s FÜHRE Anweisungen AUS	