Strukturierte Programmierung

Grundlegendes über strukturierte Programmierung
sowie die Methoden:
Pseudocode, Programmablaufplan und Struktogramme

Stand 15.09.2017

Was versteht man unter strukturierter Programmierung?

1. Funktionale Strukturierung des Problems
   (Datenflussdiagramm, SADT, ...)

2. Strukturierung der Daten
   (Data Dictionary, Syntaxdiagramm, ...))

3. Strukturierung des Programmablaufs

Regeln

1. Beschränkung auf wenige Kontrollstrukturen

- Sequenz
- Selektion, Auswahl
- Iteration, Wiederholung, Schleife
- Aufruf anderer Algorithmen (Unterprogramme)

2. Single Entry / Single Exit - Prinzip

- Jedes Konstrukt hat genau **ein** Eingang und genau **ein** Ausgang.
- => eine Vereinfachung des Programmablaufs.
- Sprünge (GOTO) werden dadurch vermieden.

Regeln

3. Programmentwicklung durch schrittweise Verfeinerung (top-down)

Bei komplexen Programmstrukturen ist es jederzeit möglich, Bereiche auf separate Diagrammseiten auszulagern.

Dies entspricht der Bildung von Unterprogrammen bzw. Unterprogrammaufrufen.

Pseudo-Code

Pseudocode

- textuelle, semiformale Darstellungsform in **Anlehnung** an problemorientierte Programmiersprachen.
- Pseudocodes sind **nicht normiert**.

Kontrollstrukturen

Syntax und Wortsymbole von Programmiersprachen

Anweisungen

entweder verbale Formulierungen

oder mehr oder weniger programmiersprachliche Notationen

Pseudocode (Eigenschaften)

Vorteile:

- Präzise
- Übersichtlich
- Änderungsfreundlich
- leicht in Quellcode übertragbar

Bestandteile:

bei Programmentwurf: keine geschweiften Klammern

- Schlüsselworte (beschreiben den Ablauf)
- natürliche Sprache (formuliert die Aktionen)
- Einrückungen (verdeutlichen die Struktur)

Ziel:

Pseudocode soll lesbar und verständlich sein für

- Personen ohne (besondere) Programmierkenntnisse
  (Auftraggeber, Vertrieb, Projektmanager, …)
- Entwickler und Programmierer

Pseudocode (Regeln)

Allgemein:

- Unbedingt einrücken!
- Schlüsselworte auffällig scheiben
  (**fett** oder *kursiv* oder ***beides*** oder mit GROßBUCHSTABEN)

WICHTIG für die Entwurfsphase::

- In der Entwurfsphase gibt es **kein** „***i = i + 1***“
- Nicht zu sehr an Programmiersprachen orientieren
- Unnötige Klammerungen, Semikolons usw. vermeiden
- Klare und saubere Sätze formulieren
  (Substantive, Verben, Adjektive: => besser lesbar)

TIPP:

- Bei der Erstellung jede 2. Zeile frei lassen für einfachere Ergänzungen bzw. Änderungen

Pseudocode: Schlüsselworte und Konstrukte

Notation:	Pascal-ähnlich	„natürliche“ Schreibweise	Beispiel
(Funktions-) Block	**BEGIN** *Name* **END** *Name*	**BEGINN** *Name* **ENDE** *Name*	**BEGINN** *Datenkonvertierung* **…** **ENDE** *Datenkonvertierung*
Sequenz:	Anweisung 1 Anweisung 2 Anweisung 3	Anweisung 1 Anweisung 2 Anweisung 3	**…** Lies Eingabedaten ein Konvertiere Eingabedaten Schreibe Daten in Datei **…**

Auswahl:	**IF** *Bedingung*
	THEN
	Anweisung
	ELSEIF *Bedingung*
	Anweisung
	ELSE
	Anweisung
	ENDIF
		WENN *Bedingung*
		Anweisung
		ODER WENN *Bedingung*
		Anweisung
		SONST
		Anweisung
		ENDE WENN
			WENN *Element LINIE ist*
			Lies Anfangspunkt
			Lies Endpunkt
			ODER WENN *El. PUNKT ist*
			Lies Mittelpunkt
			SONST
			Lies Mittelpunkt
			ENDE WENN

Pseudocode: Schlüsselworte u. Konstrukte (2)

Notation:	Pascal-ähnlich	„natürliche“ Schreibweise	Beispiel
Mehrfach-auswahl	**CASE** *Variable* **OF** Konst 1 : Anweisung Konst 2 : Anweisung . . . Konst n : Anweisung **ELSE** Anweisung **ENDCASE**	**FALLS** *Variable* **IST** **Wert 1:** Anweisung **Wert 2:** Anweisung . . . **Wert N:** Anweisung **SONST** Ausnahme-Anweisung **ENDE-FALLS**	**FALLS** *Wichtigkeit* **IST** **error:** Meldung ist Fehler Fehler ausgeben **info:** Meldung ist Info **…** **SONST** Ausgabe „unbekannte W.“ **ENDE-FALLS**
Aufruf von Routinen:	**CALL** *UP*(argList)	**FÜHRE** *UP-Name*(argL) **AUS**	**FÜHRE** *create*(a,b, c) **AUS**
Schleifen (fest):	**FOR** *Zähler* in *Bereich* Anweisungen **ENDFOR**	**WIEDERHOLE** von *N* bis *M* Anweisungen **ENDE-WIEDERHOLE**	**WIEDERHOLE** von *N* bis *M* **FÜHRE** calc(x) **AUS** prüfe Ergebnis **ENDE-WIEDERHOLE**

Pseudocode: Schlüsselworte u. Konstrukte (3)

Notation:	Pascal-ähnlich	„natürliche“ Schreibweise	Beispiel

Schleifen: („Kopfgesteuert“)	**WHILE** *Bedingung*
	  Anweisungen
	ENDWHILE
		SOLANGE *Bedingung*
		  Anweisungen
		ENDE-SOLANGE
			SOLANGE *A kleiner als B ist*
			  gib den Wert von B aus
			  berechne neuen Wert von B
			ENDE-SOLANGE

Schleifen: („Fußgesteuert“)	**REPEAT**
	  Anweisungen
	UNTIL *Bedingung*
		WIEDERHOLE
		  Anweisungen
		BIS *Bedingung*
			WIEDERHOLE
			  berechne neuen Wert von B
			  gib den Wert von B aus
			BIS *A gleich groß ist wie B*

Spezialfall "Endlosschleife":	**FOREVER**
	  Anweisung 1
	  Anweisung 2
	BREAK
	END		

Pseudocode: Beispiel

Aufgabe: Erstellung eines Unterprogramms zur Erfassung von Firmenadressen

Setzen Sie folgende Anforderungen aus einem Lastenheft in Pseudocode um:

1. Ein Programm zur Verwaltung von Firmenadressen basiert zur Vereinfachung für das vorliegende Problem auf den Daten *Firmeneintrag* und *Firmenkurznamen*.
2. Bei der Ersterfassung einer Adresse muss geprüft werden, ob diese bereits in der Firmendatei vorhanden ist. Wenn nicht, so soll ein Firmenkurznamen vergeben werden und ein neuer Firmeneintrag in der Firmendatei vorgenommen werden.
3. Sollte bereits ein Firmeneintrag vorhanden sein, so kann man diesen ändern und dann neu eintragen.
4. Bei der Löschung eines Firmeneintrags muss darauf geachtet werden, dass es Kundeneinträge in der Kundendatei geben kann, die den Firmenkurznamen enthalten. Dann muss der Benutzer darauf hingewiesen werden, dass erst alle entsprechenden Kundeneinträge geändert werden müssen.
5. Das Programm soll benutzerfreundlich implementiert werden (hohe Interaktivität).

Pseudocode: Beispiel (Lösung)

„Hauptfunktion“

Beginn Adresserfassungsunterprogramm

FALLS Funktion **ist**

Ersterfassung: *Erfassen und Prüfen der Firmendaten*

Änderung: *Firmeneintrag lesen, anzeigen und ändern (lassen)*

Löschung: *Firmeneintrag löschen*

ENDE-FALLS

Ende Adresserfassungsunterprogramm

Bemerkung: hier ist ebenso der Aufruf von Unterprogrammen möglich

Pseudocode: Beispiel (Lösung(2))

BEGINN „*Erfassen und Prüfen der Firmendaten*“

Erfassen der Firmendaten.

Prüfen, ob Firma bereits vorhanden durch Vergleich des neuen Firmennamens mit den vorhandenen Firmennamen in der Firmendatei

WENN Firma neu ist
Einen Firmenkurznamen vergeben;
Neuen Firmeneintrag in der Firmendatei vornehmen;
SONST
Firmeneintrag anzeigen und überprüfen
WENN Änderung vorgenommen wurden
Geänderten Firmeneintrag in Firmendatei eintragen;
ENDE-WENN
ENDE-WENN

ENDE „*Erfassen und Prüfen der Firmendaten*“

Pseudocode: Beispiel (Lösung(2))

BEGINN „*Firmeneintrag lesen, anzeigen und ändern*“

Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen u. anzeigen
WENN Änderung vorgenommen wurden
Geänderten Firmeneintrag in Firmendatei eintragen
ENDE-WENN

ENDE „*Firmeneintrag lesen, anzeigen und ändern*“

BEGINN „*Firmeneintrag löschen*“

Prüfen, ob es Kundeneinträge in der Kundendatei gibt, die den Firmenkurznamen enthalten.
WENN *Kundeneinträge* einen *Firmenkurznamen* enthalten
Hinweis ausgeben, dass erst alle entsprechenden Kundeneinträge geändert werden müssen.
Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen, anzeigen und nach Bestätigung löschen.
ENDE-WENN

ENDE „ *Firmeneintrag löschen*“

Pseudocode: Beispiel (Lösung)

```
begin Adresserfassungsunterprogramm
case Funktion is
   when      Ersterfassung =>  /* Erfassen und Prüfen der Firmendaten */
                        Erfassen der Firmendaten.
                        Prüfen, ob Firma bereits vorhanden durch Vergleich des neuen Firmennamens mit den vorhandenen
                        Firmennamen in der Firmendatei
                        if  Firma ist neu then
                                    Vergabe eines Firmenkurznamens;
                                    Neuen Firmeneintrag in der Firmendatei vornehmen;
                        else
                                    Firmeneintrag anzeigen und überprüfen
                                    if  Änderung vorgenommen then
                                                Geänderten Firmeneintrag in Firmendatei eintragen;
                                    end if
                        end if
   when      Änderung =>  /* Firmeneintrag lesen, anzeigen und ändern (lassen) */
                        Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen u. anzeigen
                        if  Änderung vorgenommen then
                                    Geänderten Firmeneintrag in Firmendatei eintragen
                        end if
   when      Löschung =>   /* Firmeneintrag löschen */
                        Prüfen, ob es Kundeneinträge in der Kundendatei gibt, die den Firmenkurznamen enthalten.
                        if  Kundeneinträge einen Firmenkurznamen enthalten then
                                    Hinweis ausgeben, dass erst alle entspr. Kundeneinträge geändert werden müssen.
                                    Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen,
                                    anzeigen und nach Bestätigung löschen.
                        end if;
end case
end Adresserfassungsunterprogramm
```

Pseudocode: Beispiel (Lösung)

```
Beginn Adresserfassungsunterprogramm
FALLS Funktion ist
        Ersterfassung:   /* Erfassen und Prüfen der Firmendaten */
                Erfassen der Firmendaten.
                Prüfen, ob Firma bereits vorhanden durch Vergleich des neuen Firmennamens mit den
                        vorhandenen Firmennamen in der Firmendatei
                WENN  Firma neu ist
                        Einen Firmenkurznamen vergeben;
                        Neuen Firmeneintrag in der Firmendatei vornehmen;
                SONST
                        Firmeneintrag anzeigen und überprüfen
                        WENN  Änderung vorgenommen wurden
                                Geänderten Firmeneintrag in Firmendatei eintragen;
                        ENDE-WENN
                ENDE-WENN
        Änderung:   /* Firmeneintrag lesen, anzeigen und ändern (lassen) */
                Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen u. anzeigen
                WENN  Änderung vorgenommen wurden
                        Geänderten Firmeneintrag in Firmendatei eintragen
                ENDE-WENN
        Löschung:   /* Firmeneintrag löschen */
                Prüfen, ob es Kundeneinträge in der Kundendatei gibt, die den Firmenkurznamen enthalten.
                WENN  Kundeneinträge einen Firmenkurznamen enthalten
                        Hinweis ausgeben, dass erst alle entsprechenden Kundeneinträge geändert werden
müssen.
                        Firmeneintrag anhand des Firmenkurznamens aus der Firmendatei lesen, anzeigen
                                und nach Bestätigung löschen.
                ENDE-WENN
ENDE-FALLS
Ende Adresserfassungsunterprogramm
```

Pseudocode: Beispiel zur Integration

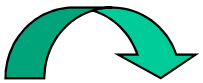
Integration eines Pseudocodes in Quellcode

```
Beginn Ratensparen
        Daten einlesen (Monatsbetrag, Laufzeit, Zins_pro_Jahr, Boni)
                unter Beruecksichtigung der Grenzwerte
        Zinsdauer aus Laufzeit in Monaten berechnen
        solange Monat < Zinsdauer
                Zins fuer vergangenen Monat berechnen
                Guthaben um berechneten Zins erhoehen
                wenn zwoelf Monate um sind
                        aktuellen Bonus berechnen
                        Guthaben um berechneten Bonus erhoehen
                        aktuelle Daten ausgeben
                ende-wenn
                Guthaben um eingezahlten neuen Betrag erhoehen
        ende-solange
ende Ratensparen
```

Pseudocode: Beispiel zur Integration in Quellcode

```
/* Ratensparen */

#include <stdio.h>

main()
{       double Zins_pro_Jahr, Zins_pro_Monat, Monatsbetrag, Bonus[25];
        double  Guthaben = 0.0, Zinsguthaben_Gesamt = 0.0, Zinsguthaben_M = 0.0,
                Gesamtbonus = 0.0, Bonusguthaben = 0.0, Zinsdauer = 0.0;
        int i, Laufzeit, Monat, Jahr = 0;

        /* Daten einlesen (Monatsbetrag, Laufzeit, Zins_pro_Jahr, Boni)
           unter Beruecksichtigung der Grenzwerte */
        printf("******************* Ratensparen ********************");
          printf("\ngeben Sie folgendes ein: \n\nMonatsbetrag [DM]: ");
        scanf("%lf", &Monatsbetrag);
        do{
                printf("\nLaufzeit [5 bis 25 Jahre]: ");
                scanf("%d", &Laufzeit);
        }while( Laufzeit < 5 || Laufzeit > 25 );

        printf("\nZins_pro_Jahr: ");
        scanf("%lf", &Zins_pro_Jahr);
        Zins_pro_Monat = Zins_pro_Jahr / 1200;

        for( i=0 ; i<Laufzeit ; i++ ){
                printf("\nJahres-Bonus(Jahr %d): ",i+1);
                scanf("%lf", &Bonus[i]);
                Bonus[i] *= .01;
        }
```

Pseudocode: Beispiel zur Integration in Quellcode (2)



```
    /* Zinsdauer aus Laufzeit in Monaten berechnen  */
    Zinsdauer = Laufzeit * 12 ;
    Guthaben = Monatsbetrag;

    for( Monat = 1 ; Monat <= Zinsdauer ; Monat++ ) /* solange Monat < Zinsdauer */
    {
            /* Zins fuer vergangenen Monat berechnen */
            Zinsguthaben_M = Guthaben * Zins_pro_Monat;
            printf("\nMonat(%d):\tGH=%8.2f,\tZGH=%8.2f", Monat, Guthaben, Zinsguthaben_M);

            /* Zinsguthaben akkumuliert ... */
            Zinsguthaben_Gesamt += Zinsguthaben_M;

            /* Guthaben um berechneten Zins erhoehen */
            Guthaben += Zinsguthaben_M;

            /* wenn zwoelf Monate um sind */
            if( Monat  && ( ! ( Monat % 12 ) ) )
            {
                    /* aktuellen Bonus berechnen und Guthaben um ber. Bonus erhoehen */
                    if( Bonus[Jahr] > 0.0 )  Bonusguthaben = Monatsbetrag * 12 * Bonus[Jahr];
                    Guthaben = Guthaben + Bonusguthaben;
                    Gesamtbonus += Bonusguthaben;
                    Jahr++;
                    /*  aktuelle Daten ausgeben  */
                    printf("\nJahr(%d):\t%8.2f\t%8.2f\t%8.2f\t%8.2f\n", Jahr, Guthaben,
                            Bonusguthaben, Zinsguthaben_Gesamt, Gesamtbonus );
            }
            /* Guthaben um eingezahlten neuen Betrag erhoehen */
            Guthaben = Guthaben + Monatsbetrag;
    }
    printf("\n%8.2f\t%8.2f\t%8.2f\t%8.2f\n",  Guthaben, Bonusguthaben, Zinsguthaben_Gesamt, Gesamtbonus );
} /* ende Ratensparen */
```

Beispielaufgabe *Ampelschaltung*

Gegeben sei folgende vereinfachte Problembeschreibung einer Ampelschaltung:

- Wenn die Ampel grün ist, dann darf man fahren.
- Wenn die Ampel rot ist, dann muss man anhalten.
- Wenn die Ampel gelb ist und der Bremsweg ausreicht, dann muss man anhalten, reicht er nicht aus, muss man fahren.

Modellieren Sie diese Problembeschreibung mit Pseudocode

Lösungsvorschlag für Beispiel *Ampelschaltung*

BEGINN Ampelschaltung
    WENN die Ampel grün ist:
        dann darf man fahren.
    ODER-WENN die Ampel rot ist:
        dann muss man anhalten.
    ODER-WENN die Ampel gelb ist:
        WENN der Bremsweg ausreicht:
            dann muss man anhalten,
        ODER-WENN der Bremsweg nicht ausreicht:
            dann muss man fahren.
        ENDE-WENN
    ENDE-WENN
ENDE Ampelschaltung

Beispielaufgabe *Geldwechselautomat*

- Ein alter, vereinfachter Geldwechselautomat gibt nach der Eingabe eines Geldbetrags Geldstücke aus nach dem Prinzip der *minimalen Anzahl bei maximaler Diversifizierung*.
- Die maximal wechselbare Geldmenge liegt bei 10 DM
- die minimal wechselbare Geldmenge liegt bei 2 DM
- ansonsten wird eine Meldung angezeigt.
- Dabei können Münzen im Wert von jeweils 1 bis 5 DM ausgegeben werden.
- Es wird vorausgesetzt, dass sich genügend Wechselgeld im Automaten befindet.

Lösungsvorschlag für Beispiel *Geldwechselautomat*

BEGINN Geldwechselautomat
    WENN die eingeworfene Geldmenge 10 DM ist:
        5 DM ausgeben
        FÜHRE „*5 DM wechseln und ausgeben*“ **aus**
    ODER-WENN die Geldmenge 5 DM ist:
        FÜHRE „*5 DM wechseln und ausgeben*“ **aus**
    ODER-WENN die Geldmenge 2 DM ist:
        1 DM ausgeben
        1 DM ausgeben
    ENDE-WENN
ENDE Geldwechselautomat

BEGINN „*5 DM wechseln und ausgeben*“
    2 DM ausgeben
    2 DM ausgeben
    1 DM ausgeben
ENDE „*5 DM wechseln und ausgeben*“

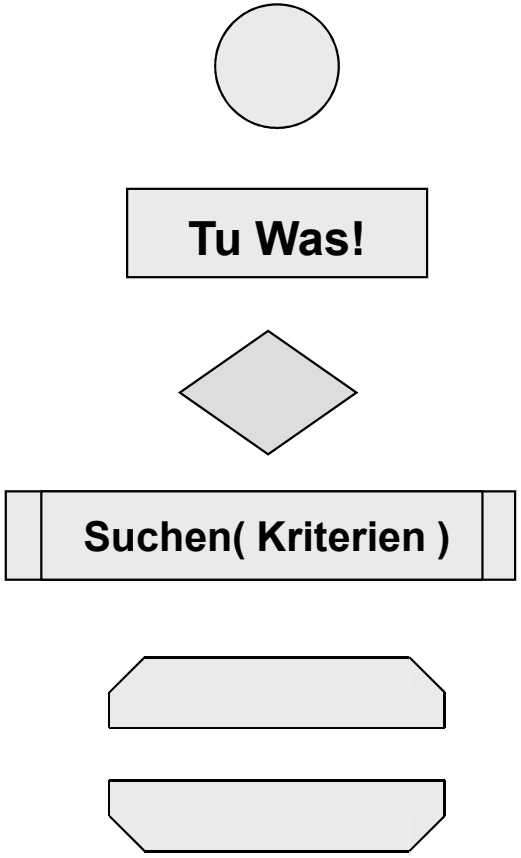
Aufgaben zu Pseudocode

- Erweiterte Ampelschaltung
- Fußballtipprunde

Programmablaufplan (PAP)

Programmablaufplan

- betont das dynamische Ablaufverhalten eines Programms (SW)
- besteht aus lediglich 6 Symbolen, die durch Linien bzw. Pfeile (Flüsse) miteinander verbunden werden



Start / Ende / Fortsetzung

Anweisung

Abfrage

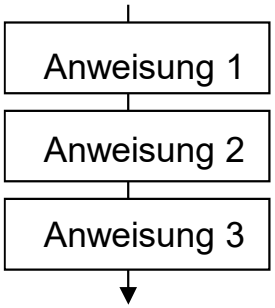
Aufruf (Unterprogramm, Algorithmus)

Schleifenanfang

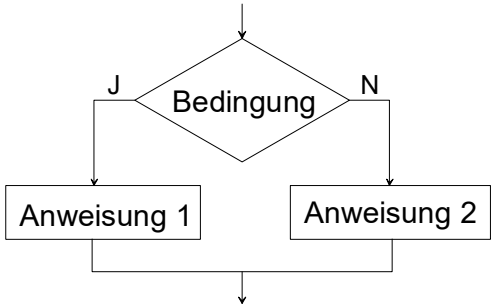
Schleifenende

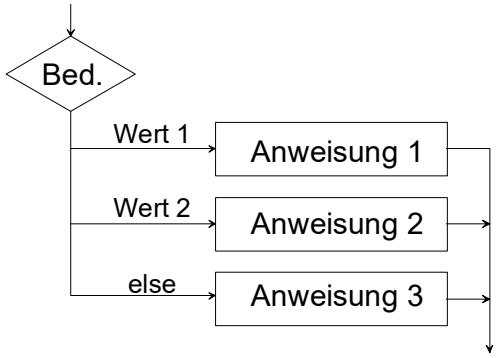
Programmablaufplan (Kontrollstrukturen)

Sequenz:

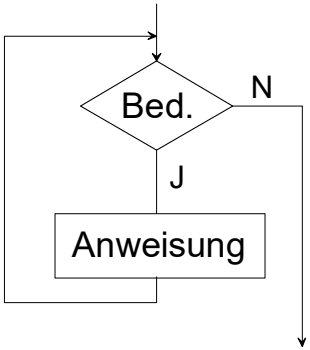


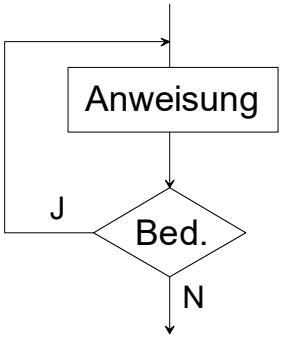
Selektion:

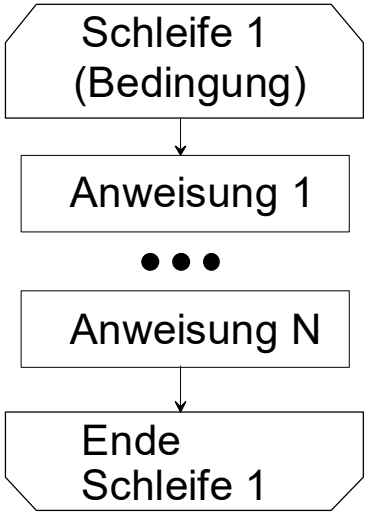




Iteration, Schleife:







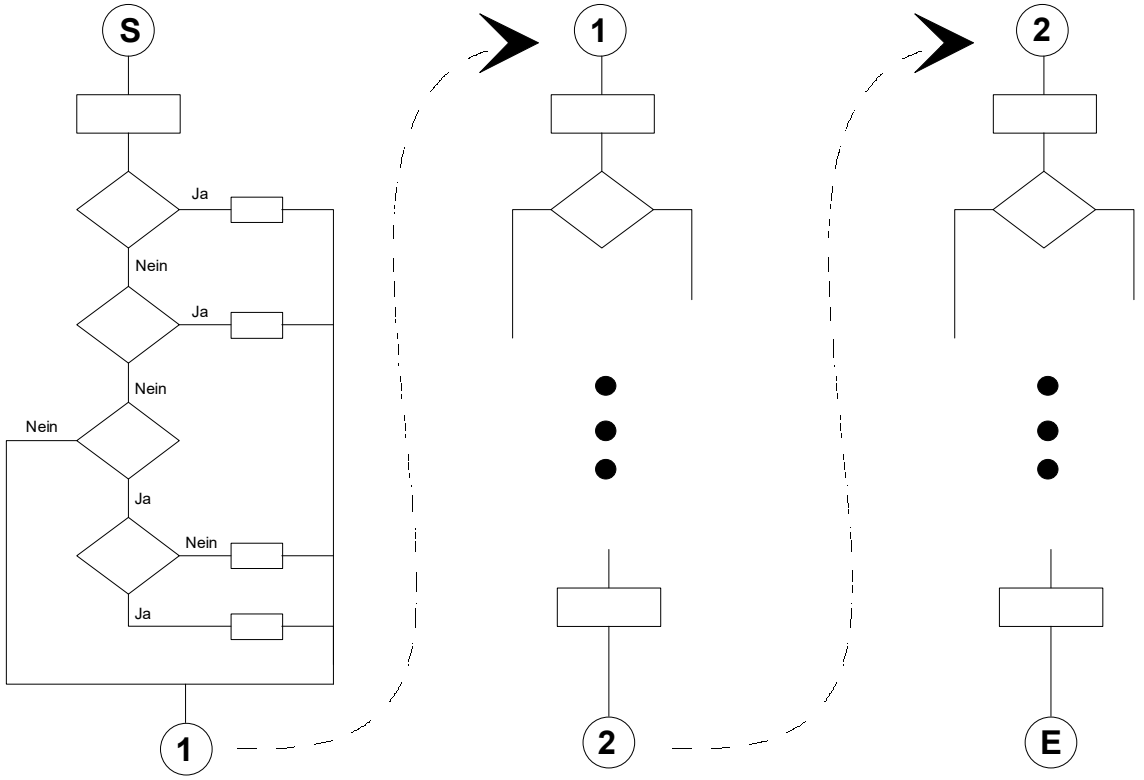
PAP (Grundregeln zum Zeichnen der Diagramme)

- **wesentliche** Flussrichtung von (links) oben nach (rechts) unten
- Anzahl der Anweisungen (Rechtecke) möglichst minimieren
- **Anweisungen (Rechtecke):**
  Eingang: **oben** / (links bei CASE-Konstrukt)
  Ausgang: **unten** / (rechts bei CASE-Konstrukt)
- **Abfragen (Raute):**
  Eingang: immer oben
  Ausgang: links / rechts / unten
- Linien: möglichst kreuzungsfrei

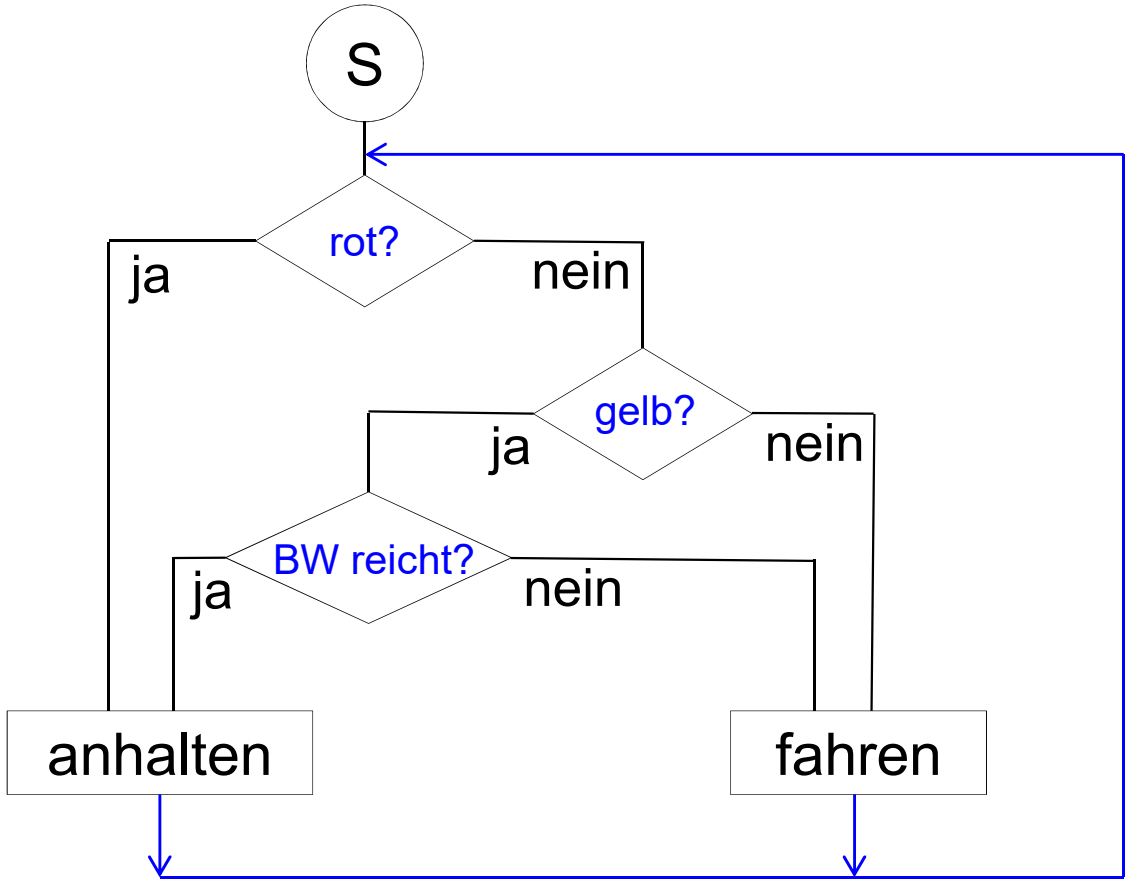
PAP (Grundregeln (2) zum Zeichnen der Diagramme)

Aufteilung bei komplexen Anwendungen

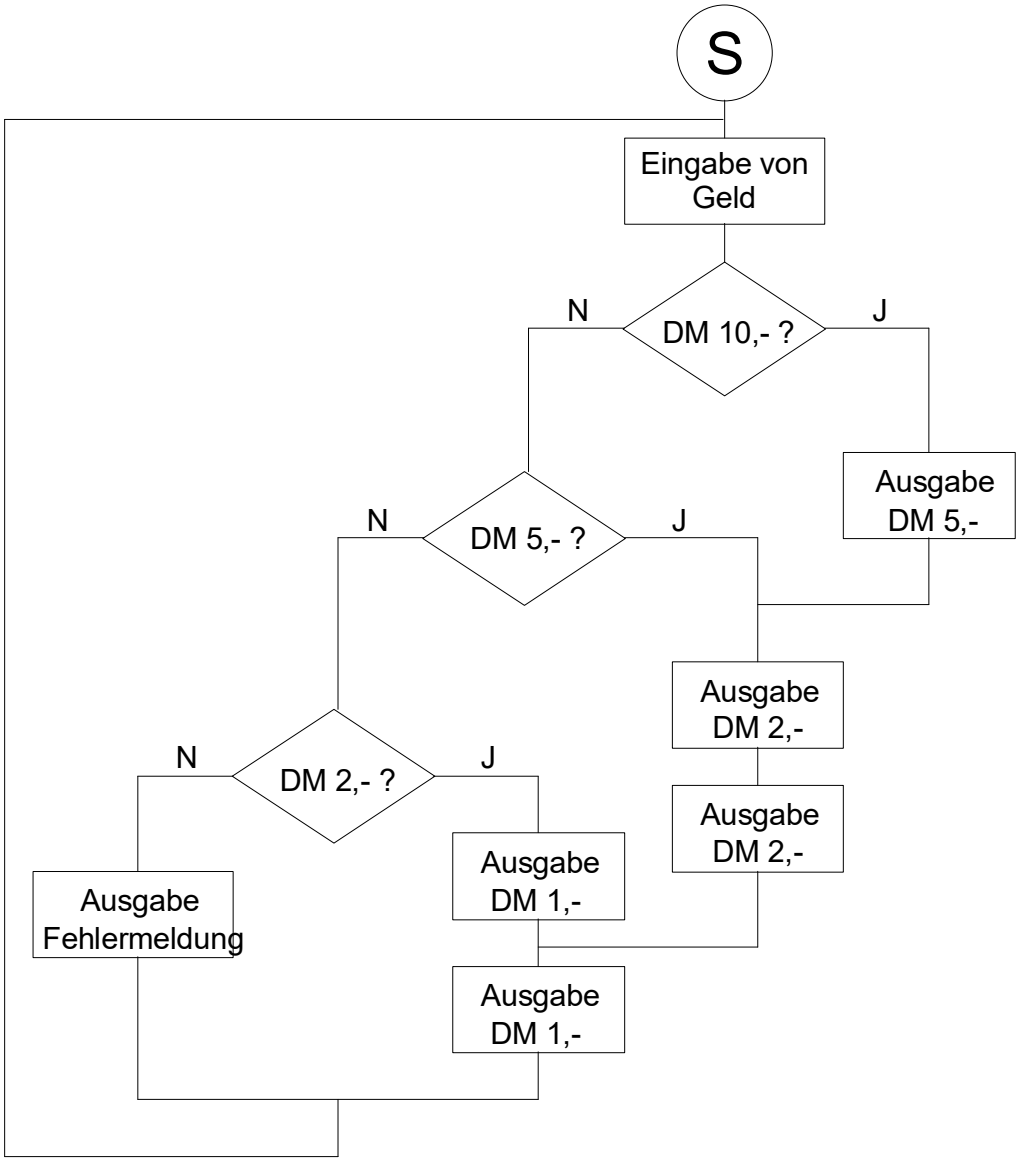
- Modularisieren (fast immer möglich), d.h. Unterprogrammaufrufe verwenden, diese möglichst auf ein separates Blatt Papier zeichnen
- Aufteilung mit Fortsetzung (ausnahmsweise):



PAP (Beispiel Ampelschaltung)



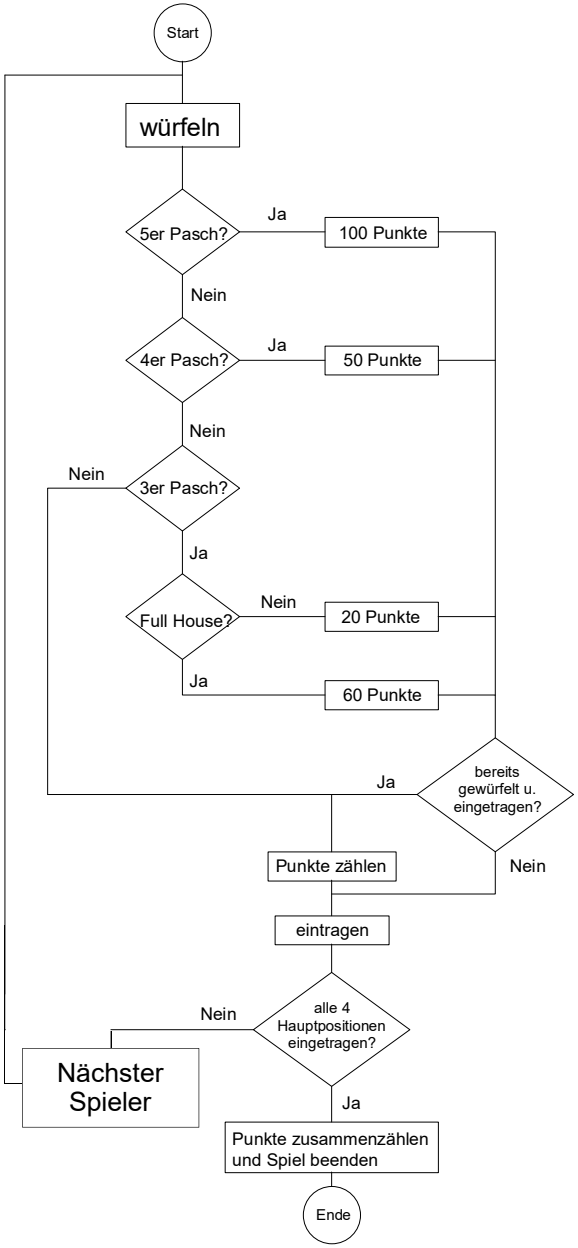
PAP (Beispiel Geldwechselautomat)

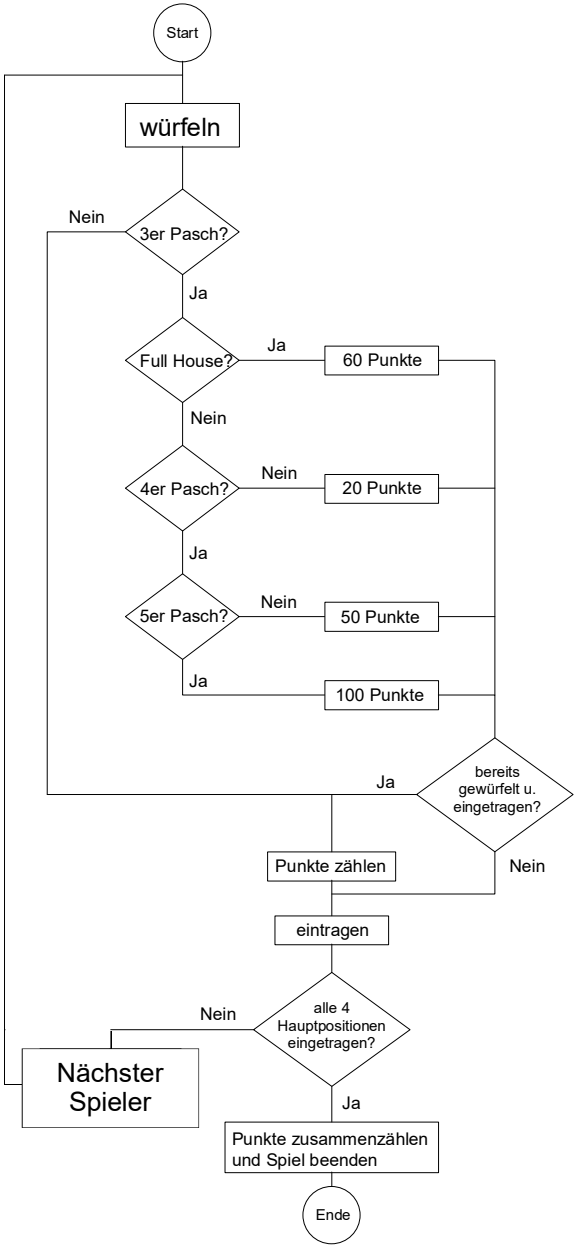


Aufgaben zu PAP

- Erweiterte Ampelschaltung
- Würfelspiel

PAP (Beispiellösung Würfelspiel)



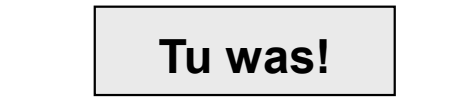


Struktogramm
(Nassi-Shneiderman-Diagramm)

Struktogramme

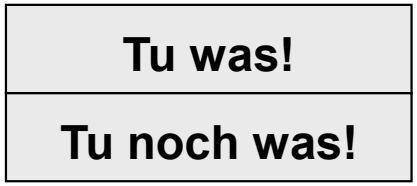
- betonen die Struktur eines Programms (SW)
- lehnen sich sehr stark an die strukturierte Programmierung an.
  ⇨ Sie werden auch Strukturdiagramme genannt
- sind normiert (DIN 66261)
- bestehen aus einzelnen Rechteck-Elementen unterschiedlicher Bedeutung, die zu einem Diagramm zusammengesetzt werden
  - diese Rechtecke können intern weiter unterteilt werden
  - Sie werden ausschließlich von oben „betreten“ und unten verlassen

Struktogramm (Anweisungen)

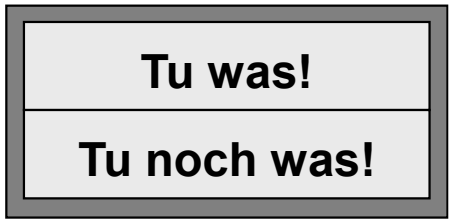


Anweisung

leere Anweisung (als Platzhalter)



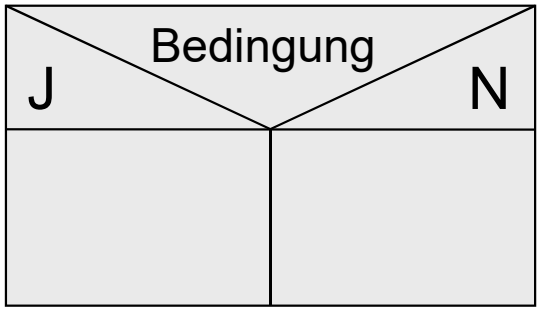
Folge von Anweisungen



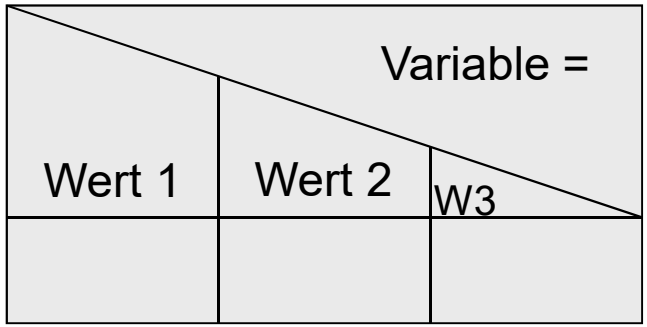
(Prozedur-)Blockbildung

Aufruf eines Unterprogramms

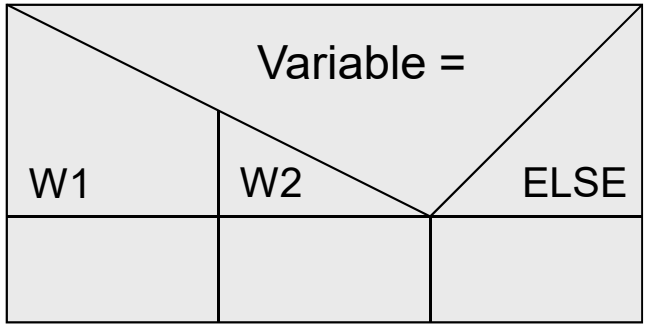
Struktogramm (Auswahlkonstrukte)



Auswahl



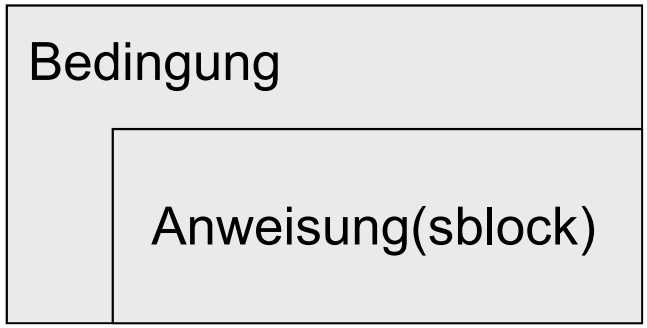
Mehrfachauswahl



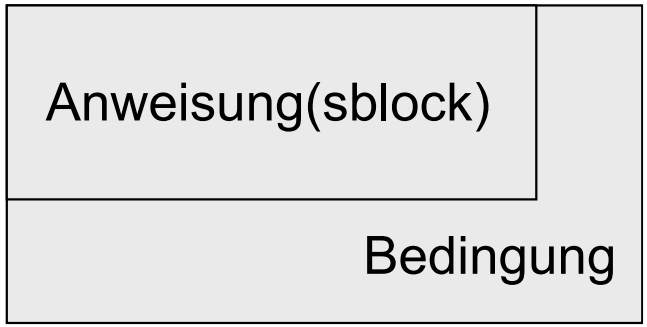
Mehrfachauswahl mit ELSE

Struktogramm (Schleifen und BREAK)

Abbruchanweisung für das Konstrukt ***N***

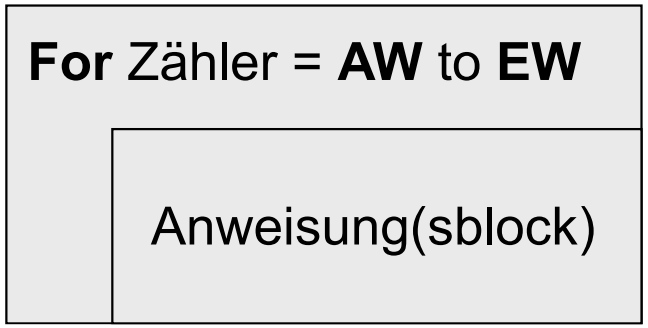


kopfgesteuerte Schleife

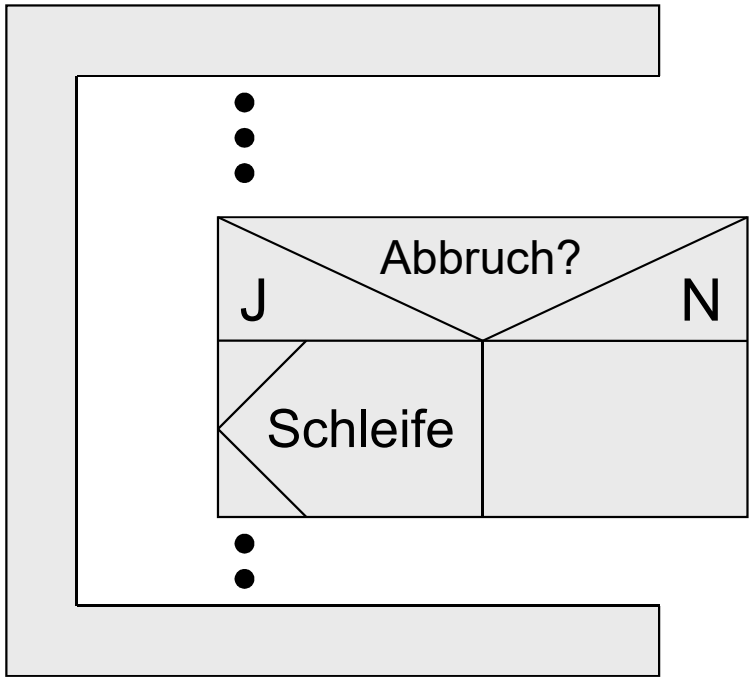


fußgesteuerte Schleife

Struktogramm (Schleifen (2))



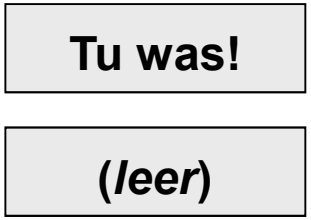
Wiederholung mit fester Wiederholungszahl (keine DIN-Norm!)

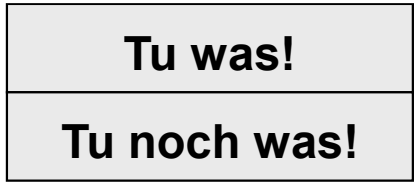


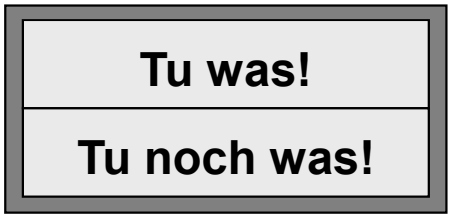
Wiederholung ohne Bedingungsprüfung

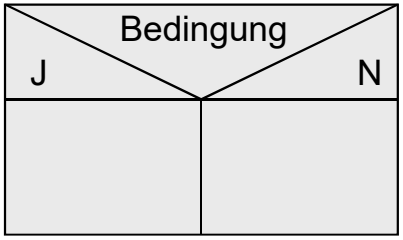
- sog. FOREVER-Schleife
- Spezialfall der kopfgesteuerten Schleife

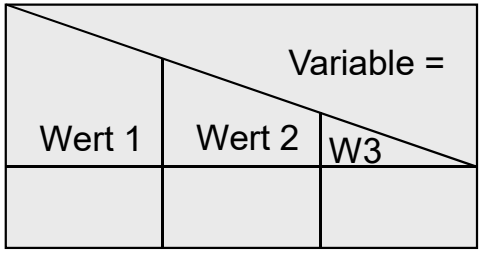
Struktogramm (Konstrukte, Gesamtübersicht)

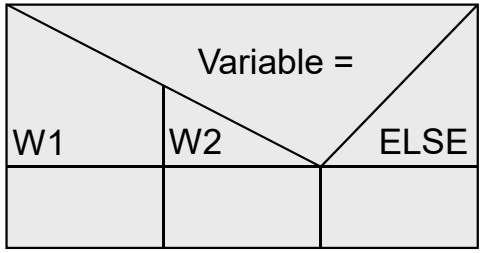


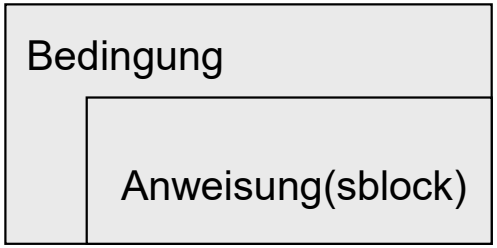


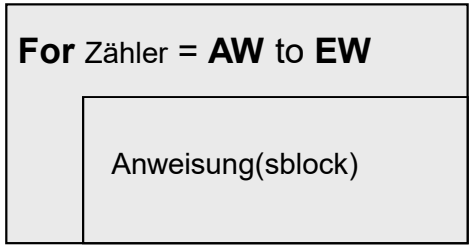


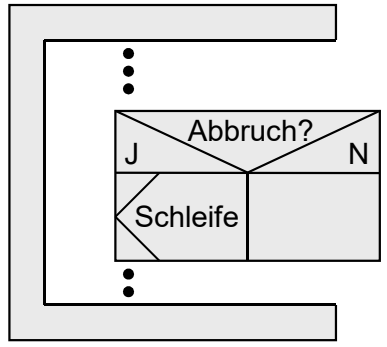








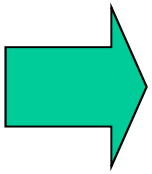




Struktogramm (Vor- und Nachteile)

Vorteile:

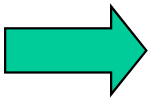
- Zwang zur Strukturierung

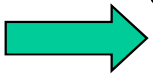
  - übersichtliche und damit verständliche Dokumentation
  - Erleichterung der späteren Wartung
  - lehnen sich sehr stark an die strukturierte Programmierung an

Nachteile:

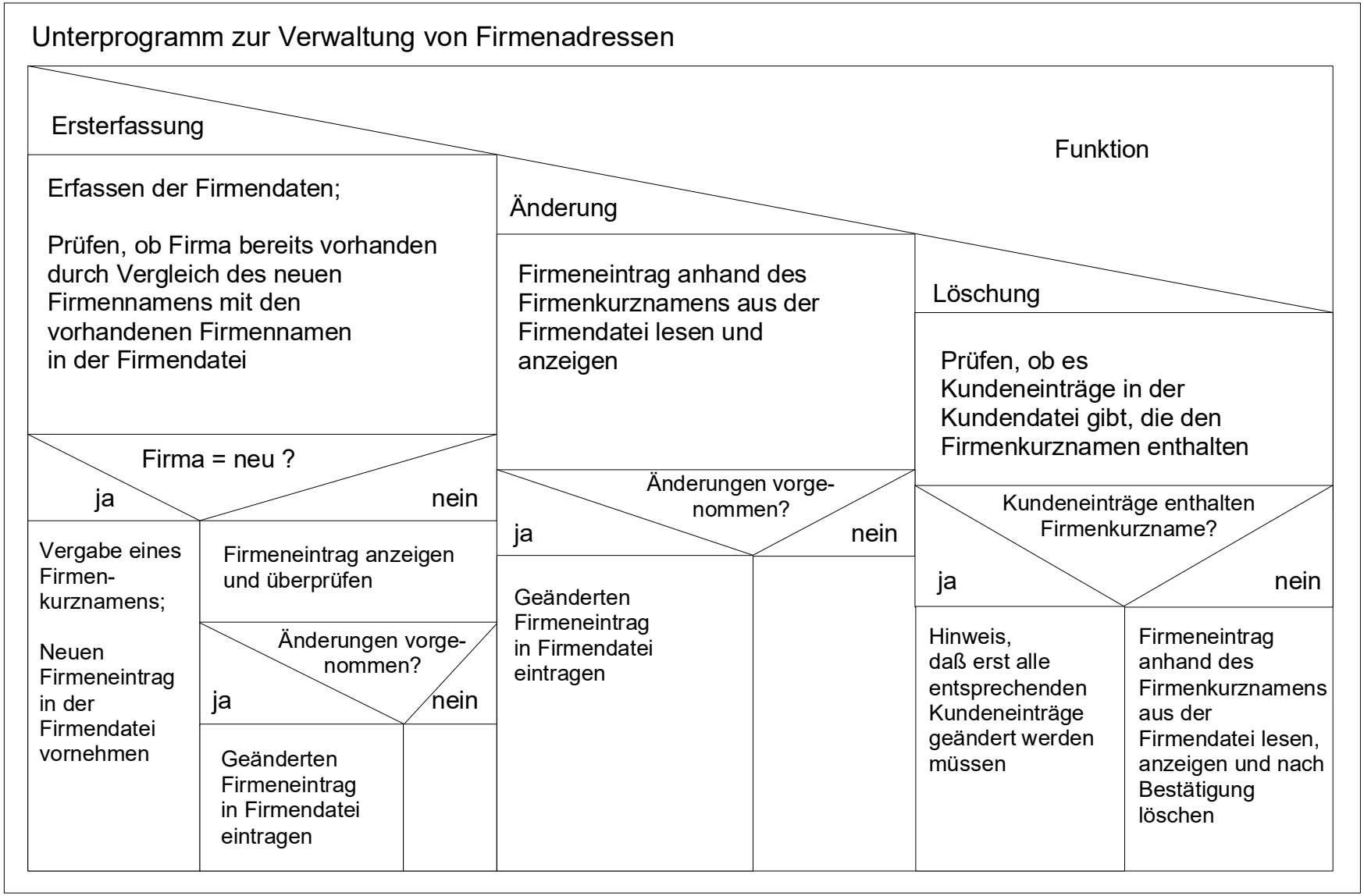
- Zur Strukturierung des Problems ist eine detaillierte Problemkenntnis nötig, was in der Analysephase selten der Fall ist

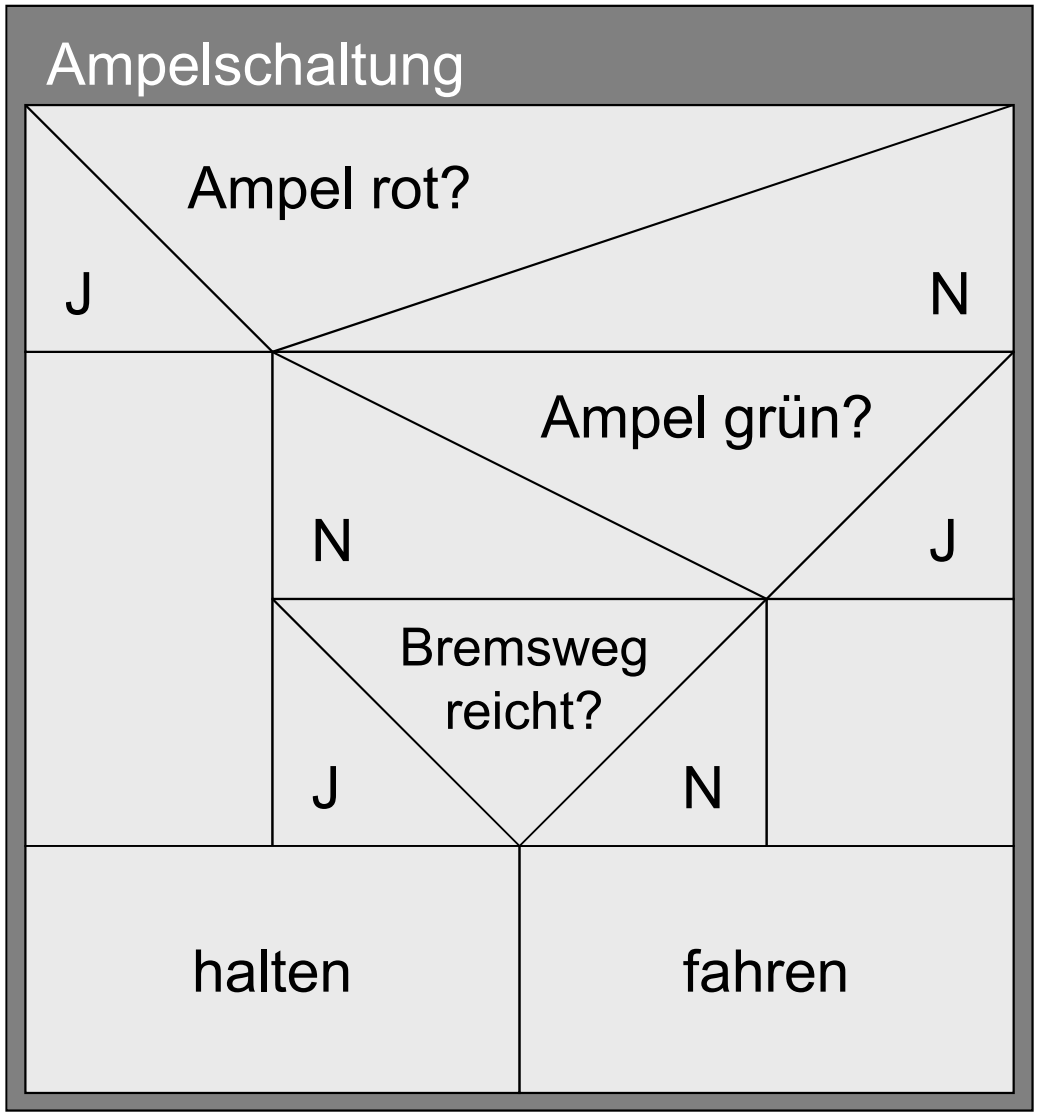
  Struktogramme sind eigentlich erst möglich, wenn das Problem bereits strukturiert ist

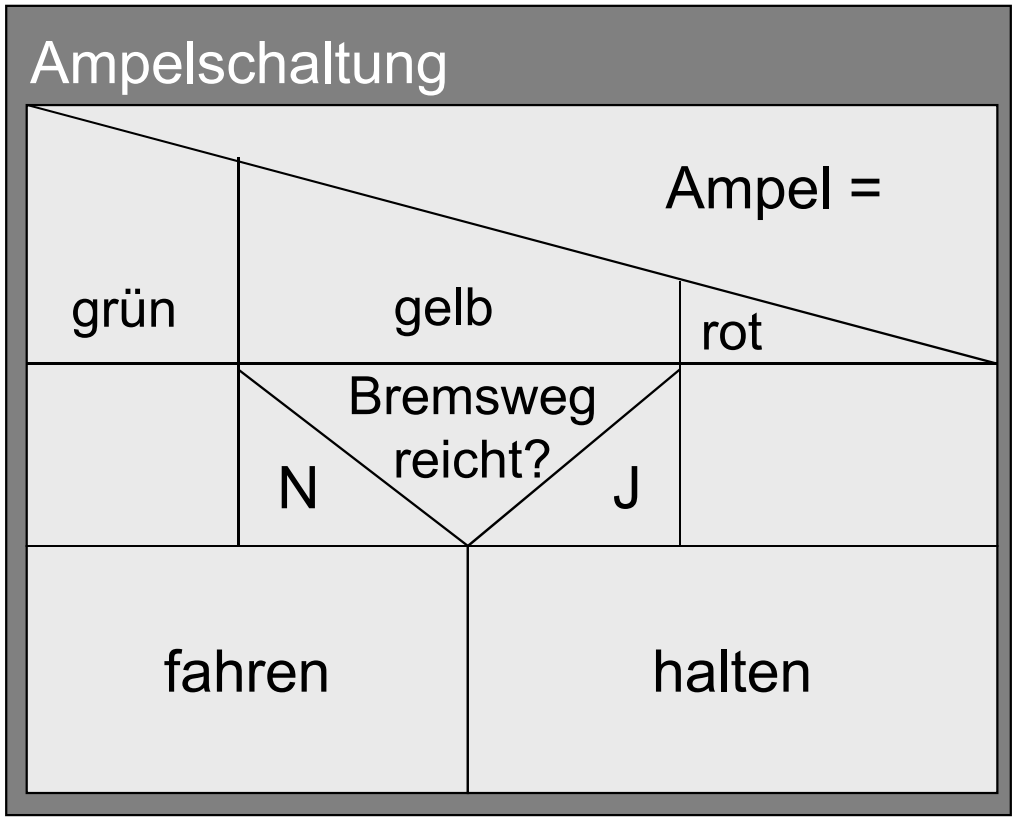
- schlechte Änderbarkeit der Diagramme aufgrund ihrer graphischen Beschaffenheit.  Einsatz eines CASE-Tools.

Struktogramm (Beispiel)



Struktogramm (Lösungsvorschläge Ampelschaltung)





Struktogramm (Lösungsvorschläge Geldwechselautomat)