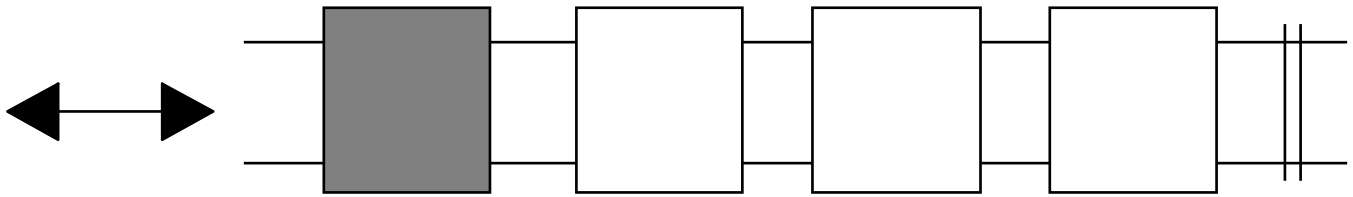
Spezielle lineare Felder

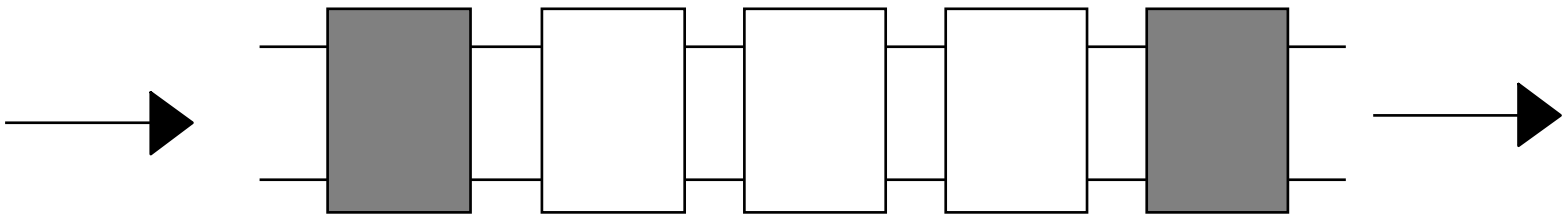
1. Stapel:



→ Neues Element kann nur an das zuletzt abgelegte angehängt werden;

Entnahme erfolgt nach dem LIFO (Last-In-First-Out) Prinzip

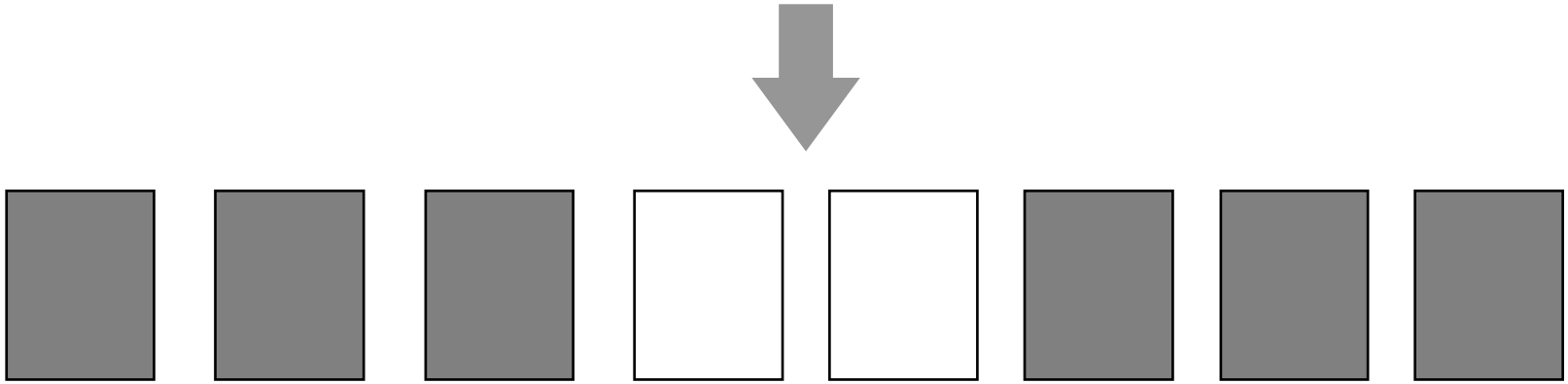
2. Schlange:



→ Entnahme nach FIFO (First-In-First-Out) Prinzip,

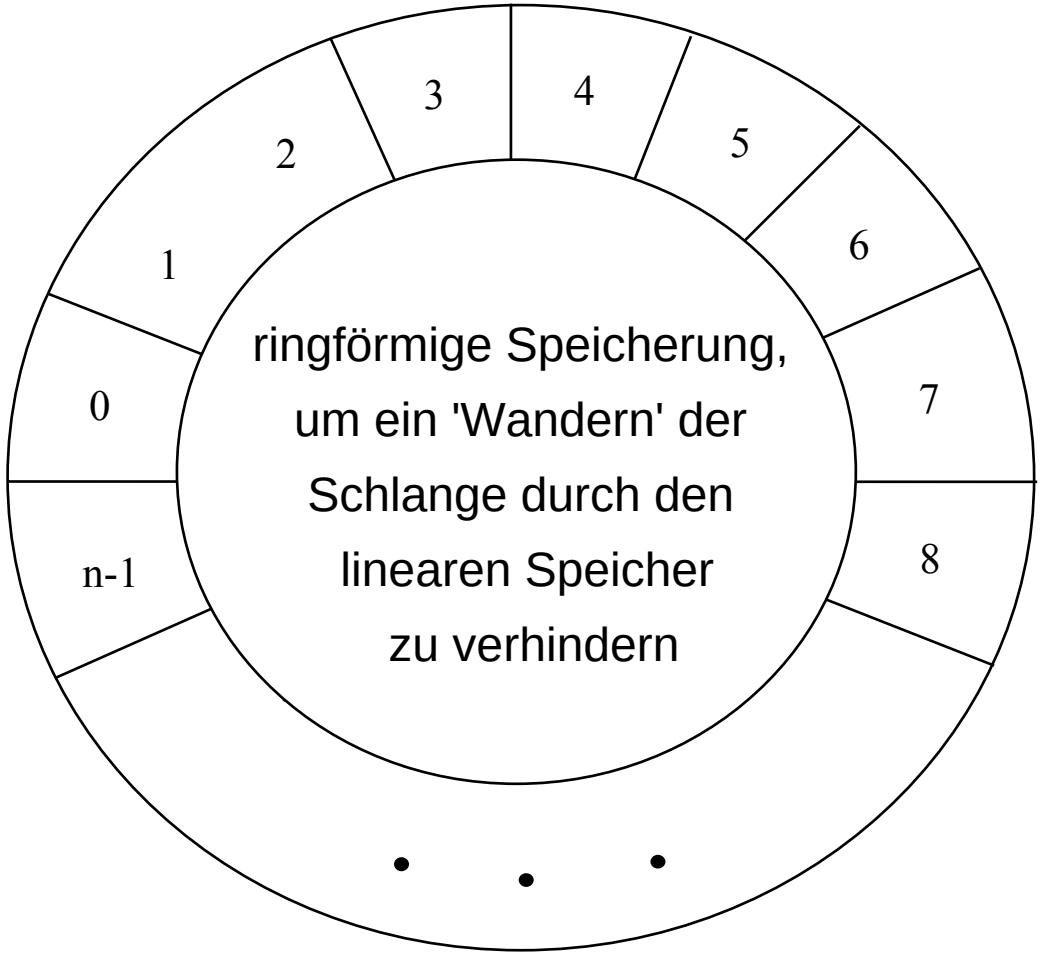
Hinzufügen durch Anhängen

3. Zwei Stapel:



→ Zwei Stapel werden sequentiell mit den Zugangsfeldern zueinander gespeichert.
Stapel läuft erst über, wenn beide Stapel keinen Platz mehr finden.

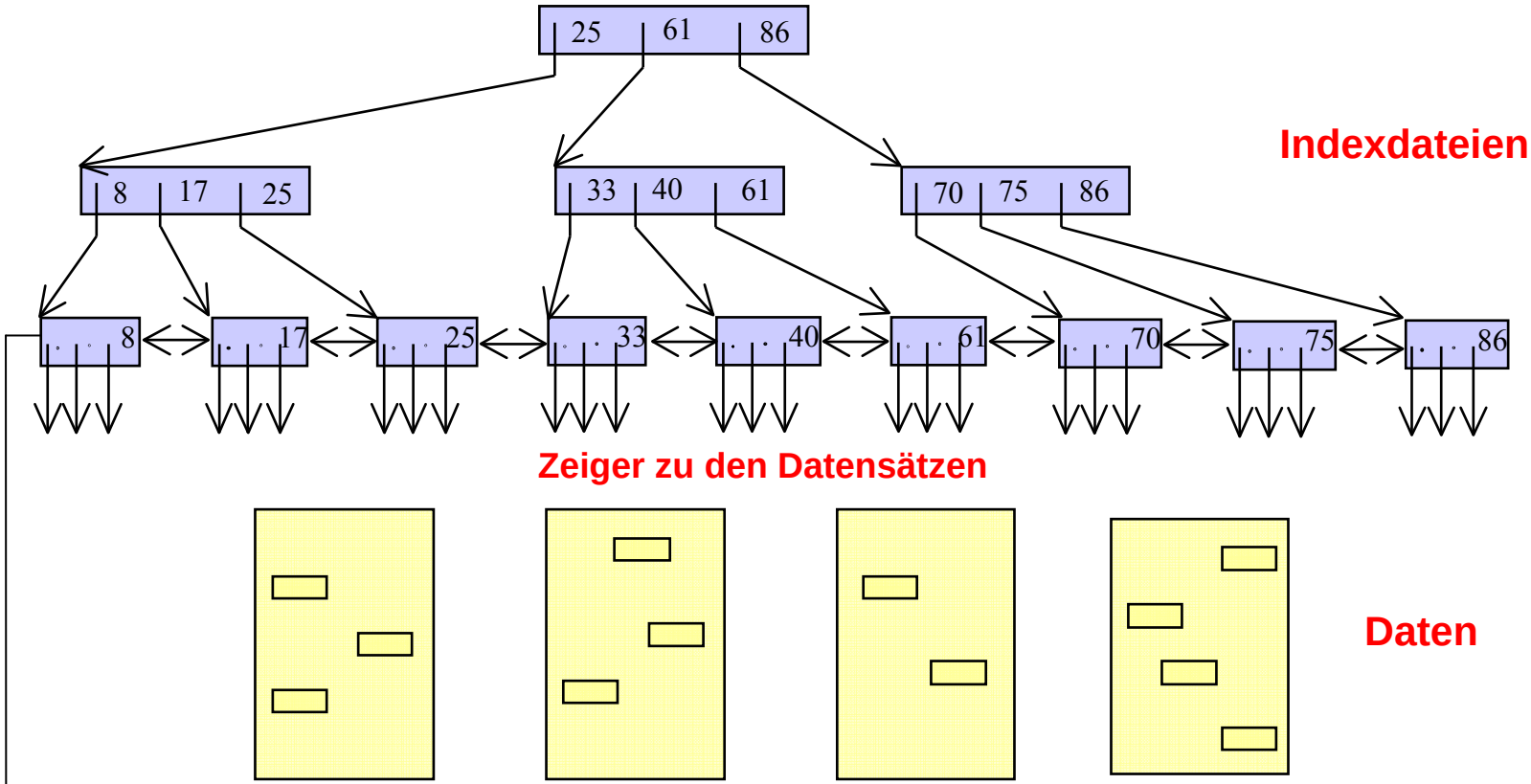
4. Schlange mit ringförmiger Speicherform:



Indexdateien

- Index-sequentielle Speicherung, d.h. für jede Datei wird eine Indexdatei und eine Überlaufdatei angelegt.
- Indexdatei enthält Suchschlüssel des Datensatzes aus der Primärdatei und Zeigerfeld, welches die Speicheradresse des Datensatzes angibt.
- Sucht man nach einem Satz mit einem entsprechenden Schlüssel, so genügt es, die Indexdatei (viel kleiner als die Primärdatei) zu durchsuchen.
- Indexdateien über Indexdateien bei großen Datenbeständen: Indexdatei enthält nur jeden k-ten (k > 1) Schlüssel (Mehrwegebaum).

Dreistufiger Index eines 3-Wege-Baumes



Auf der unterster Ebene, dem sog. Blattknoten, sind Zeiger gespeichert, die zu den eigentlichen Daten weisen.

5.3 Darstellung von Funktionen durch Algorithmen

Algorithmen

→ Funktionen benutzen Algorithmen.

Definition Algorithmus:

Exakte Beschreibung einer Verarbeitungsfolge, die ohne zusätzliche Erläuterungen von einer Maschine ausgeführt werden kann.

Beispiel für Algorithmus:

Aufgabe besteht in der Verschmelzung zweier Listen mit reellen Zahlen. In beiden Ausgangslisten sind die Zahlen nicht-fallend geordnet. Dies soll in der Liste nach Verschmelzung erreicht werden.

-- **Kopf und Deskriptor**

Algorithmus *merge*

Input: Two sequences of reals, e = (e[1],...,e[n]) and f = (f [1],...,f [m]), both sorted in non-decreasing order.

Output: Sequence g = (g[1],...,g[n + m]) in which all elements are arranged in non-decreasing order.

-- Methode

```
begin
i := 1; j := 1; k := 1;  -- initialization of counters
while     (i <= n) and (j <= m) do
                    -- the while loop merges elements of sequences e and f into g;
                    -- the loop is executed until all elements of one of the sequences
                    -- are merged
          begin
          if        e[i] < f [j]
          then      begin g[k] := e[i]; i := i + 1; end
          else      begin g[k] := f [j]; j := j + 1; end;
          k := k + 1;
          end;
if        i <= n    -- not all elements of sequence e have been merged
then      for x := i to n do g[k + x - i] := e[x]
else
          if        j <= m    -- not all elements of sequence f have been merged
          then      for x := j to m do g[k + x - j] := f [x];
end;
```

Algorithmen

- **Block:** Mehrere Anweisungen - eingebettet durch "begin" und "end"

- **Anweisung:**

  - Block
  - Zuweisung
  - Else- oder Case-Operation
  - Schleife ("for", "while", "repeat... until", ein allgemeiner loop)
  - call eines anderen Algorithmus,
  - Abbruch einer Schleife (exit loop) etc.

- **Case-Anweisungen**: Zerlegen die Ausführung eines Algorithmus in mehrere Äste, abhängig vom Wert der Variablen.

- **Loop-Anweisungen**: "for all a <aus> M do ...", oder
  " while M <nicht_leer> do ...".

- **Exit-Loop**: Algorithmus springt zur ersten Anweisung nach der Schleife zurück

- **Kommentare**: Durch zwei Minuszeichen (--) eingeleitet und durch Ende der Zeile (EOL) abgeschlossen.

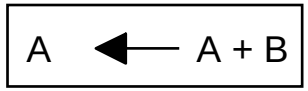
Programmablaufpläne

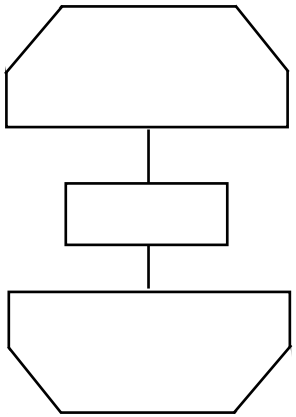
- Übersichtliche Darstellung des **Ablaufs** (Kontrollstruktur) eines **Programms** (Algorithmus) mit Programmablaufplan und Struktogramm

- **Kontrollstrukturen**:
  - Sequenz,
  - Alternative,
  - Wiederholung,
  - Rekursion und
  - Parallelität.

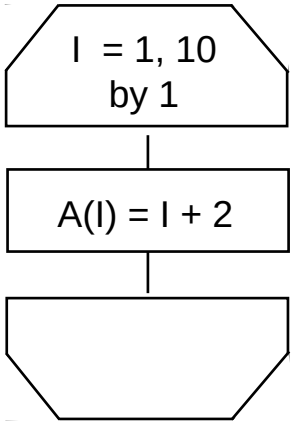
Programmablaufpläne (PAP) nach DIN 66001 (Nachteil: PAP führen oft zu unübersichtlichen und unnötig langen Darstellungen von Algorithmen):

Allgemeine Operationen:
z.B. arithmetische Berechnungen wie Addition, Multiplikation etc.

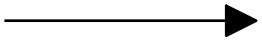




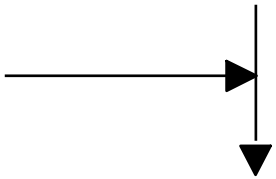
Schleifenbegrenzung: die Angaben über Bedingungen sind am Schleifenanfang oder Schleifenende anzugeben



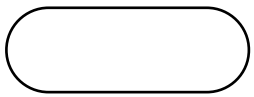
Eingabe, Ausgabe



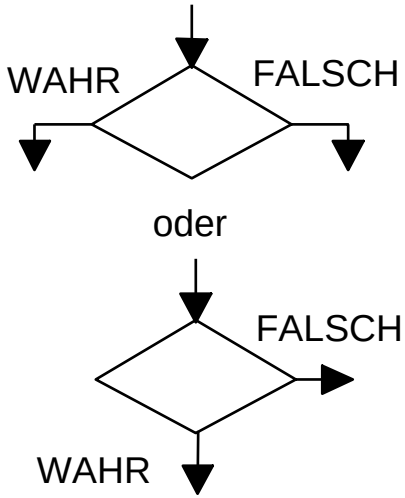
Ablauffolge



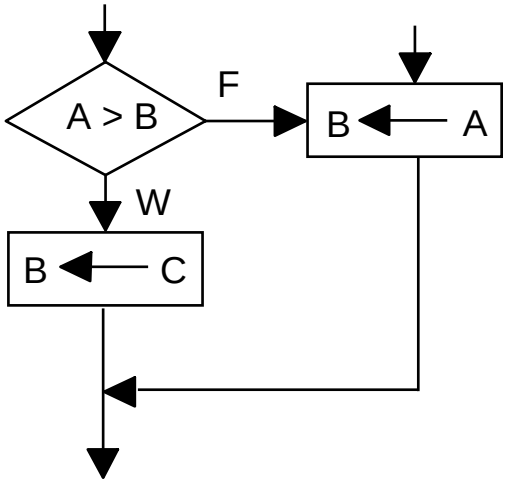
Zusammenführung

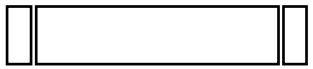


Begrenzung: bezeichnet Anfang und Ende eines Programms



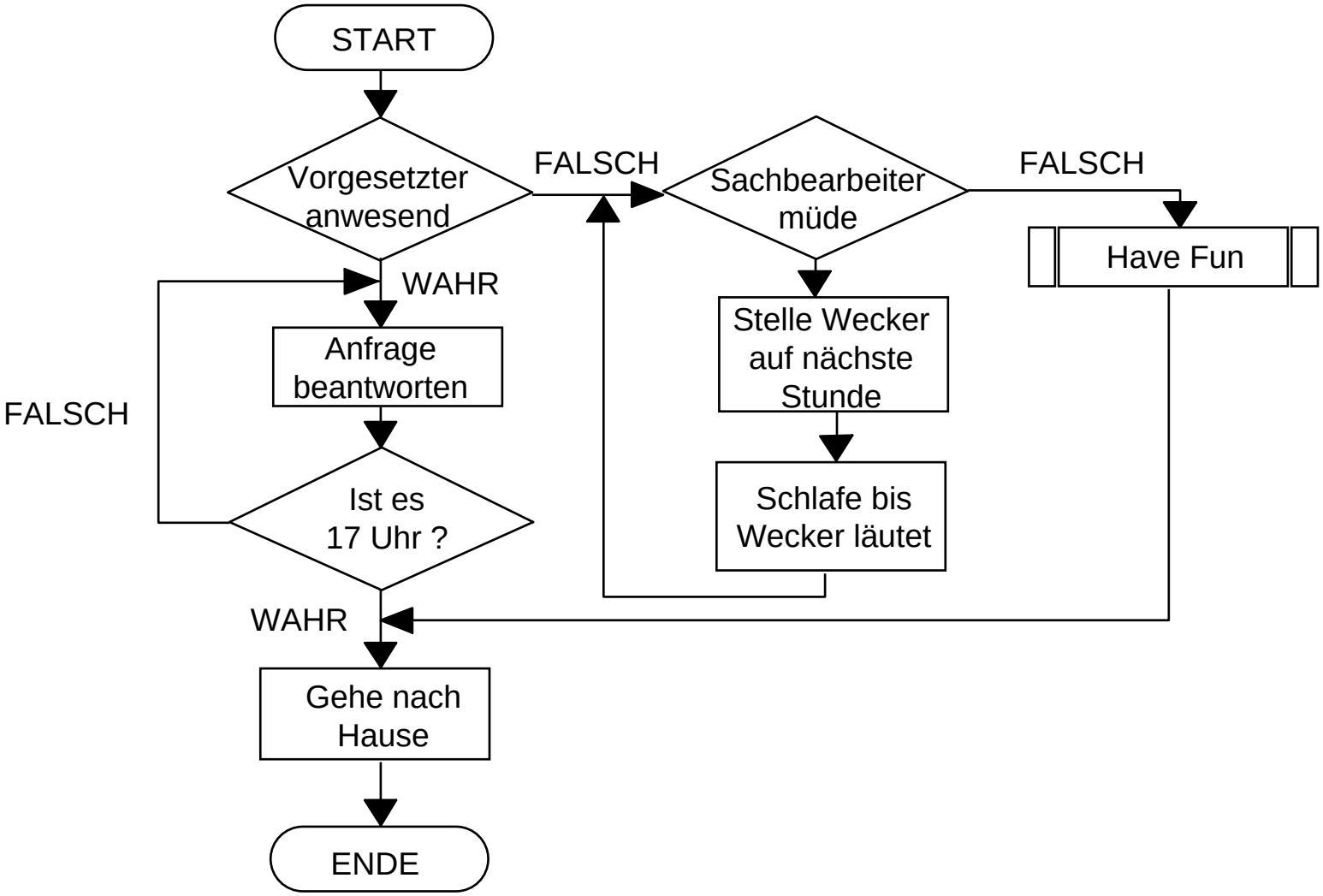
Bedingte Verzweigung:
Bedingung wird als Raute repräsentiert. Existieren nur zwei Alternativen werden ein WAHR- und ein FALSCH-Pfad unterschieden.



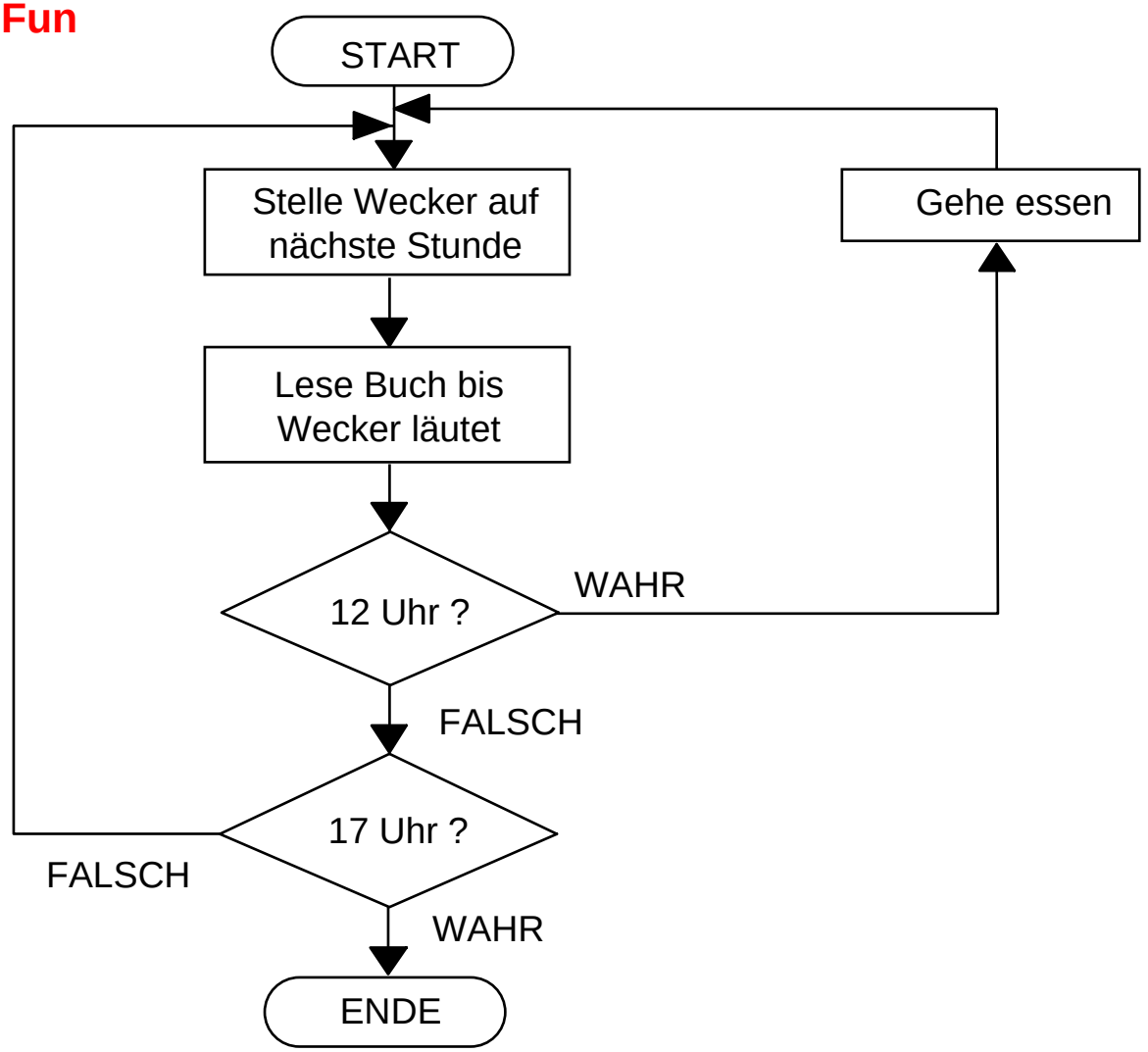


Unterprogramm: Unbedingter Sprung zu einem anderen Programmteil (Programmmodul), nach dessen Ausführung das Programm an dieser Stelle fortgesetzt wird.

Beispiel: **Tagesablauf eines Sachbearbeiters**



Beispiel: **Have Fun**



Struktogramme

→ Struktogramme greifen auf die folgenden Komponenten zurück:

- **Anweisung** zur Darstellung eines Verarbeitungsschrittes
  (z.B. Wertzuweisung, Ein-, Ausgabeanweisung, Prozeduraufruf)

- **Entscheidung** zur Darstellung der bedingten Verzweigung

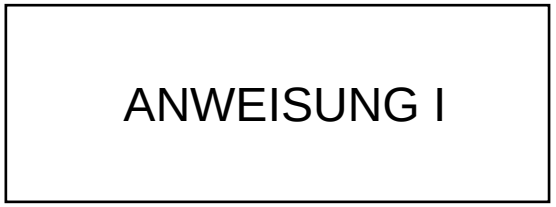
- **While Do** wenn die Ausführung der Anweisung(en) von einer *Anfangs*bedingung abhängig gemacht wird.

- **Repeat Until** wenn die Ausführung der Anweisung(en) von einer *End*bedingung abhängig gemacht wird.

- **Beginn Ende** dient zur Begrenzung zusammengehöriger Anweisungsteile (Blöcke).

1. Das Anweisungsdiagramm:

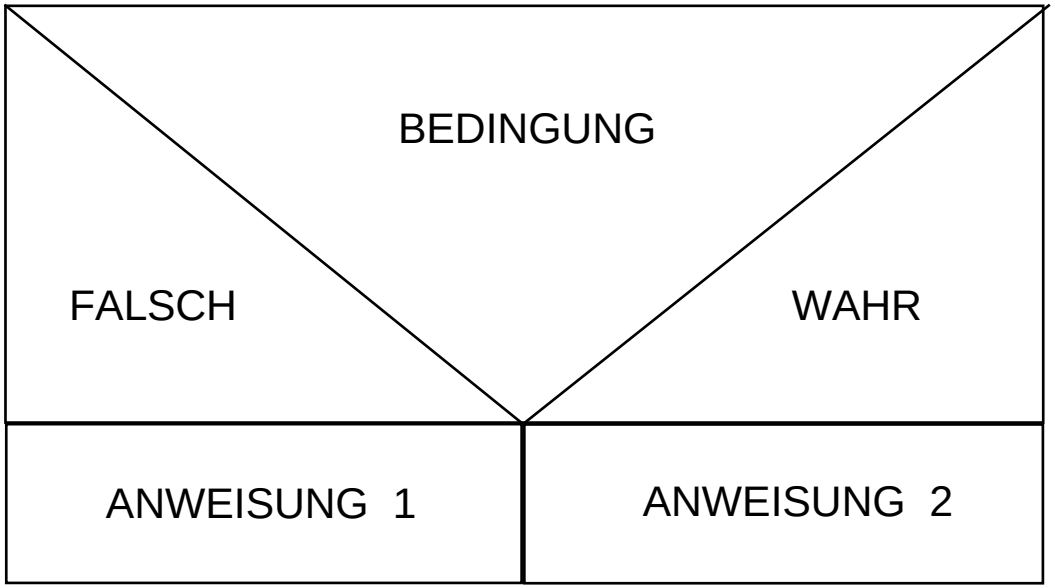
→ dient der Darstellung eines Verarbeitungsschrittes, z.B. einer Wertzuweisung, einer Ein- bzw. Ausgabeanweisung oder eines Unterprogramm-Aufrufes



2. Das Entscheidungsdiagramm:

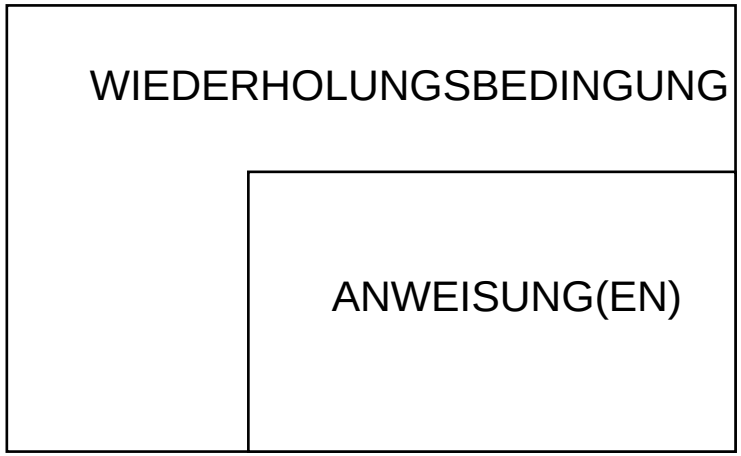
→ wird zur Repräsentation der bedingten Verzweigung verwendet.

- Zentrales Dreieck enthält Bedingung
- Dreiecke rechts und links den Wahrheitswert
- Abhängig von der Antwort wird Anweisung 1 oder 2 ausgeführt



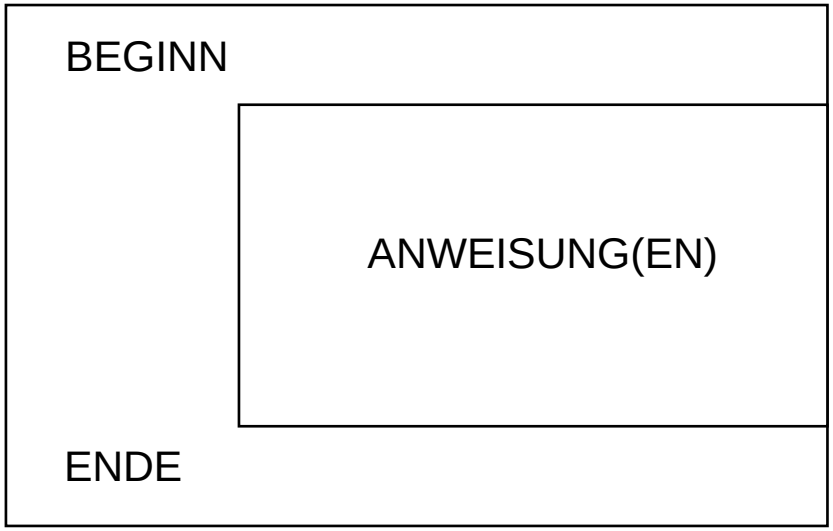
3. Das While-Do-Diagramm:

→ dient als Wiederholungssymbol, bei dem die Ausführung den Anweisungen von der einleitenden Bedingung abhängig gemacht wird. Ist die Bedingung erfüllt, werden die Anweisungen nicht mehr ausgeführt.



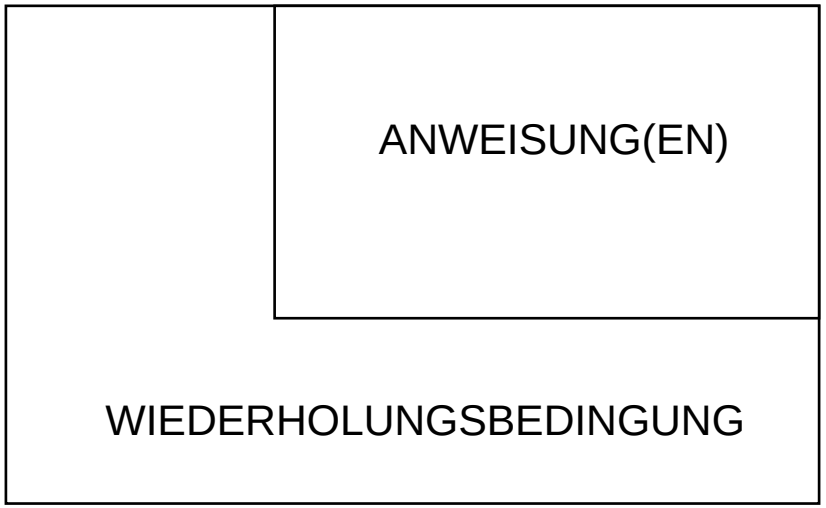
4. Das Beginn-Ende-Diagramm:

→ dient zur Hervorhebung zusammengehöriger Anweisungsteile, sog. *Blöcke.*



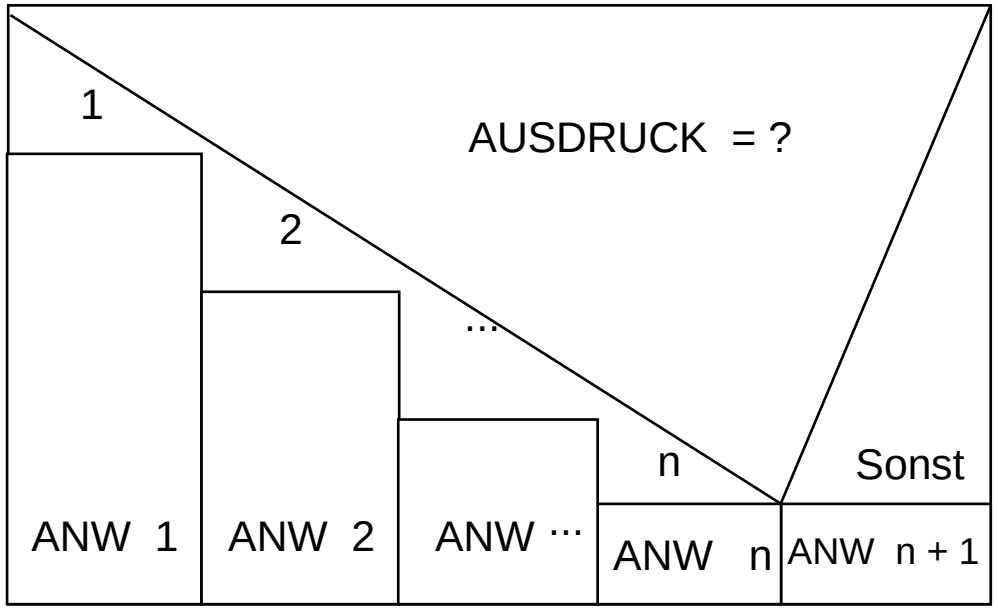
5. Das Repeat-Until-Diagramm:

→ dient als Wiederholungssymbol, bei dem die Durchführung der Anweisung(en) von einer der Endbedingungen abhängt.



6. Das Case-Diagramm:

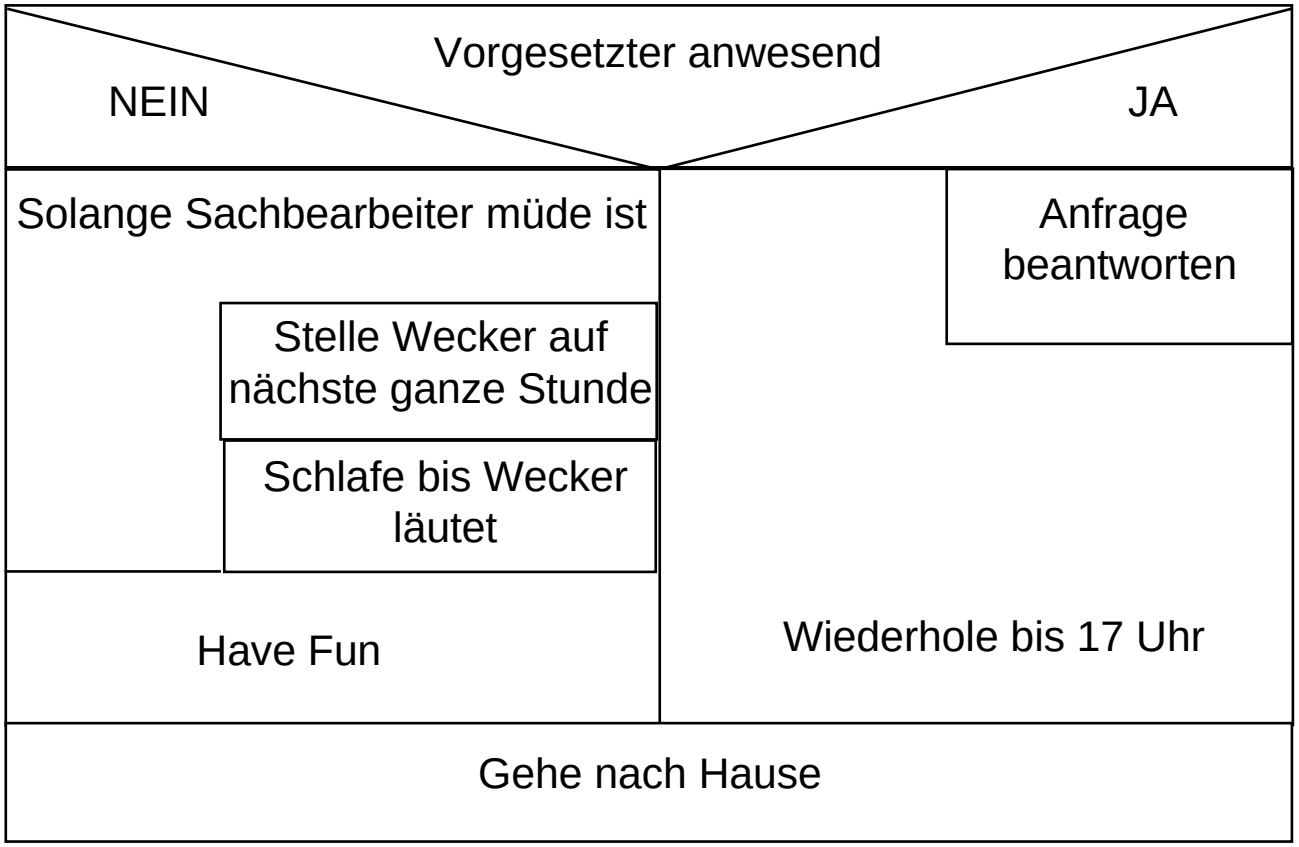
→ dient als Entscheidungskomponente für Mehrfachverzweigungen, wobei die Verzweigung vom Wert eines Ausdruckes abhängig gemacht wird.



Zusammenhang:

Ist der Wert des Ausdrucks i, so wird die Anweisung ANW 1 ausgeführt, sonst die ANW n+1.

Beispiel 1: Sacharbeiter-Struktogramm



Beispiel 2: Sacharbeiter-Struktogramm