## Anweisungen

Eine Anweisung ist eine in einer beliebigen Programmiersprache abgefaßte Arbeitsvorschrift für einen Computer.
Jedes Programm besteht aus einer bestimmten Anzahl von Anweisungen.
Wir unterscheiden einfache und strukturierte Anweisungen.
Die ***strukturierten Anweisungen***, insbesondere die Wierholungsanweisungen, kontrollieren den Programmfluß. Deshalb werden sie auch ***Kontrollstrukturen*** genannt.

## Leere Anweisung, Sprunganweisung, Methodenanweisung

- Leere Anweisung:
  besteht aus nichts und bewirkt nichts
  Beispiel: ; ;
- Sprunganweisung:
  Anweisungen werden nicht mehr in der Reihenfolge abgearbeitet, in der sie aufgeführt werden, sondern es erfolgt über die Anweisung GOTO ein Sprung zu einer im Deklarationsteil definierten Sprungmarke.
- Methodenanweisung
  OOP: Kapselung von Daten und zugehörigen Operationen (Methoden) zu einem Objekt

## Wertzuweisung

- ein Ausdruck wird ausgewertet und das Ergebnis wird einer Variablen zugewiesen
- alte Wert der Variablen wird überschrieben
- i.d.R. müssen die Variable und das Ausdrucksergebnis vom gleichen Datentyp sein
  (Ausnahme: Variable vom Typ real und Ausdruck vom Typ integer)

  (Aufgaben zu Wertzuweisungen siehe Vorlesungsbeilage S. 49)

## Prozeduranweisung

- Eine Prozedur ist eine Anzahl von Anweisungen, die mit einem Namen versehen worden sind und unter diesem Namen aufgerufen werden können.
- Beim Aufruf können von Fall zu Fall verschiedene Parameter übergeben werden.

## Verbundanweisung

- Eine Verbundanweisung faßt eine Folge von Anweisungen syntaktisch zu einer Anweisung zusammen.
- Die Verbundanweisung wird benötigt, weil an vielen Stellen eines Pascal-Programms nur die Ausführung einer einzigen Anweisung zugelassen ist.
- Eine bestimmte Anzahl von Anweisungen wird durch Umklammerung mit BEGIN und END zu einem Anweisungsblock zusammengefaßt.
- Die Anweisungen innerhalb der Verbundanweisung werden hintereinander abgearbeitet.
- Die von BEGIN und END eingeschlossene Anweisungen wird vom Compiler als zusammengehörig erkannt.

## Bedingte Anweisungen

- Die Ausführung der Anweisung nur dann, wenn eine bestimmte Bedingung erfüllt ist.
- Formulierung der Bedingung als boolescher Ausdruck. D. h. Ausdruck kann nur zwei Wert annehmen, entweder „wahr“ oder „falsch“.
- Pascal bietet zwei Arten von Kontrollstrukturen, (bedingte Anweisungen)
  - IF-Anweisung
    bedingte Verarbeitung (IF .....THEN)
    einfache alternative (IF .....THEN.....ELSE)
  - CASE-Anweisung

## IF-Anweisung

- Die IF-Anweisung ermöglicht es, Programmteile in Abhängigkeit des Auswertunsergebnisses eines logischen Ausdrucks ausführen zu lassen.
- Das Ergebnis der Bedingung bzw. des logischen Ausdrucks kann nur zwei Werte annehmen, entweder „wahr“ oder „falsch“.
- Die Ergebnisvariable ist demnach vom Typ boolean.
- Wenn der boolesche Ausdruck zutrifft, d. h. „true“ ist, wird die Anweisung nach THEN ausgeführt, hat der boolesche Ausdruck den Wert „false“, wird die Anweisung nach ELSE ausgeführt.

***1) Bedingte Verarbeitung:***

- Der ELSE-Zweig ist optional. D.h. er kann auch fehlen.
  In diesem Fall (***Bedingte Verarbeitung***) wird nach dem THEN-Zweig ein Semikolon gesetzt.
  Falls der boolsche Ausdruck „true“ ist, wird die Anweisung ausgeführt, die nach THEN folgt, ansonsten wird sie übersprungen und die nächste Anweisung im Programm abgearbeitet. (Beispielaufgabe s. 52 Vorl.-Beilage)

***2) Einfache Alternative:***

- Einfache (zweiseitige) Alternative
  Die IF-THEN-ELSE-Struktur oder einfache (zweiseitige) Alternative stellt die zweite Erscheinungsform der IF-Anweisung dar.
  (Beispielaufgabe S. 54,55,56 Vorl.-Beilage)

**Einsatz von Verbundanweisungen bei IF-Anweisungen:**

- die Schlüsselwörter THEN und ELSE haben einen Geltungsbereich von nur einer einzigen Anweisung
- sollen mehrere Einzelanweisungen im THEN- bzw. Else-Zweig ausgeführt werden, so müssen diese als Verbundanweisung geklammert werden.(D. h. mit BEGIN und END geklammert werden).
  (Beispiel S.53)

***3) Einfache (zweiseitige) Alterntive mit logischer Aussage im Bedingungsteil***

- die Ausführung einer Anweisung kann auch von der Erfüllung einer logischen Aussage im Bedingungsteil abhängig sein.
  Folgende logischen Ausdrücke sind möglich.

| | |
|---|---|
| A1 AND A2 | Aussage wahr, wenn sowohl A1 als auch A2 wahr sind. |
| A1 OR A2 | Aussage wahr, wenn entweder A1 oder A2 oder beide wahr sind |
| A1 OR NOT A2 | wahr, wenn A1 und A2 wahr sind, wenn A1 wahr und A2 falsch ist oder wenn A1 und A2 falsch sind. |
| A1 AND A2 OR NOT A1 | wahr, wenn A1 und A2 wahr sind, wenn A1 falsch und A2 wahr oder A1 und A2 falsch sind. |

**Rangfolge not, and, or:**
Diese Rangfolge muß bei der Abarbeitung einer komplexen Bedingung beachtet werden. Durch geeignete Klammersetzung kann diese Rangfolge jedoch beeinflußt werden.

***Beispiel***:

| falsch: | richtig: |
|---|---|
| .... | .... |
| VAR | |
| a,b: integer | |
| .... | |
| IF a = b AND a = 0 | IF (a=b) AND (a=0) |
| .... | .... |

Werden keine Klammern bei (a=b) und (a=0) gesetzt, würde der Compiler als erstes den Ausdruck b AND a bewerten.
Da a und b nicht vom Typ boolean sind (sondern integer), würde ein Syntaxfehler angegeben.

***4) Geschachtelte IF-Anweisung***

- Auf die Schlüsselwörter THEN und ELSE kann jeweils eine beliebige Anweisung folgen, also auch eine weitere IF-Anweisung (Geschachtelte IF-Anweisung).
- Verwendung finden diese geschachtelten IF-Anweisungen dann, wenn mehr als zwei Fälle abzuprüfen sind bzw. wenn die Ausführung einer Anweisung von mehreren Bedingungen abhängig gemacht werden sollen.
- Besonderheit:
  Ist die Anweisung nach THEN selbst wieder eine IF-Anweisung, so ergibt sich eine Mehrdeutigkeit, ob ELSE zum ersten oder zweiten IF gehört.
  Es gilt die Regel, daß ein ELSE immer zur letzten (innersten) IF-Anweisung gehört, die noch keinen ELSE-Teil hat.
  (Beispielaufgaben S. 57,58,59)

## CASE-Anweisung

- Sie ist mit der IF-Anweisung verwandt und kann ggfls. durch sie ersetzt werden.
  Sie wird auch mehrfache Alternative genannt.
- Bedingte Anweisung
- Besteht aus dem Ausdruck (Selektor) und einer (beliebig langen) Liste von Zweigen, die durch Marken gekennzeichnet sind
- Ausführung derjenigen Anweisung, deren Marke einem Selektor entspricht
- Selektor muß ordinalen Typs sein.
- Alle CASE-Marken müssen (Konsten sein und ) vom selben Datentyp wie der Selektor sein.

- Case-Anweisung beginnt mit dem Wort CASE
- Nach der Streuerungsgröße folgt das Wort OF.
- Die CASE-Anweisung enthält weitere Anweisungen.
- Vor jeder dieser Anweisungen steht eine Marke
- Folgen nach dem Doppelpunkt mehrere Anweisungen, so sind diese mit BEGIN und END einzuschließen.
- Wie bei der IF-Anweisung kann in der CASE-Anweisung zusätzlich mit dem ELSE-Zweig gearbeitet werden.
- Die CASE-Anweisung endet mit dem Wort END.
- Jede Marke darf nur einmal innerhalb der CASE-Anweisung erscheinen.
- Für den Fall, daß die Steuerungsgröße nicht mit einer der Marken übereinstimmt, wird der ELSE-Zweig, falls vorhanden, abgearbeitet.

(Beispielaufgaben S. 60 – 64)

## Wiederholungsanweisungen

- **Kopfgesteuerte Schleifen**
  - Abbruchbedingung wird vor Ausführung der Anweisung geprüft
  - FOR-Anweisung, WHILE-Anweisung
- **Fußgesteuerte Schleife**
  - Abbruchbedingung wird nach Ausführung der Anweisung geprüft
  - REPEAT-Anweisung

Wiederholungsanweisungen legen die mehrfache Ausführung bestimmter Programmteile in Abhängigkeit von Bedingungen oder Zählvariablen fest.

**Geschlossene Schleifen:**
Ist zum Beispiel die Anzahl der benötigten Durchläufe im voraus bekannt, so wird i. d. R. die ***FOR-Anweisung*** verwendet.
Schleifen, bei denen die Anzahl der Durchläufe vorab bekannt ist, werden als „geschlossene Schleifen“ bezeichnet.

**Offene Schleifen:**
Bei „offenen Schleifen“, die mit Hilfe der ***WHILE- oder der REPEAT-Anweisung*** konstruiert werden, ist die Anzahl der Durchläufe unbekannt.
Des ist ein sog. Abbruchkriterium (Schleifenbedingung), welches das Verlassen der Schleife veranlaßt, vorzusehen.

**Abweisende Schleifen:**
Bei der „abweisenden Schleife“, die mit Hilfe der ***WHILE-Anweisung*** konstruiert wird, findet die Überprüfung des Abbruchkriteriums vor dem Schleifeneintritt statt.
Ist das Kriterium nicht erfüllt, so werden die Anweisungen nicht ausgeführt.

**Nicht abweisende Schleife:**
Bei der „nicht abweisenden“ ***REPEAT-Schleife*** erfolgt die Überprüfung des Abbruchkriteriums am Ende der Schleife, d. h. eine mindestens einmalige Ausführung der Anweisung ist garantiert.

## FOR-Anweisung (Zählschleife)

- dient zur Konstruktion von Schleifen mit fest vorgegebener Anzahl von Durchläufen (geschlossene Schleife).
- Sie wird deshalb auch Zählschleife genannt.
- Anzahl der Durchläufe wird über eine Laufvariable kontrolliert.
- Der Datentyp dieser Variablen muß ein Ordinaltyp sein.
- Laufvariable muß im Deklarationsteil definiert werden.
- Laufvariable bekommt einen Anfangswert zugewiesen.
- Nach jedem Durchlauf wird die Laufvariable automatisch auf den nachfolgenden Wert gesetzt, bis der angegebene Endwert erreicht ist.
- Ist der Anfangswert größer als der Endwert, wird Schleife nicht durchlaufen.
- Scheint eine Rückwärtszählung sinnvoll, so ist das Wort TO durch DOWNTO zu ersetzen, d. h. es wird heruntergezählt (hier sollte der Anfangswert größer sein als der Endwert).
- Folgen nach DO mehrere Anweisungen, so müssen sie mit BEGIN und END geklammert werden.

## RESÜMEE: FOR-Anweisung

- Kopfgesteuerte Schleife
- Laufveriable: ordinaler Typ
- Anzahl der Durchläufe muß im voraus bekannt sein
- Automatische Zählung der Durchläufe in der Schleifenanweisung

(Beispielaufgaben S. 66,67)

## WHILE-Anweisung

- definiert die Schleifendurchläufe in Abhängigkeit vom Zutreffen der Schleifenbedingung.
- Diese wird in Form eines booleschen Ausdrucks angegeben.
- Der boolesche Ausdruck wird am Eingang der WHILE-Schleife überprüft.
- Wenn der Ausdruck den Wert „true“ zurückliefert, wird der Anweisungsteil ausgeführt, und der Ausdruck erneut ausgewertet.
- Der Vorgang wiederholt sich, bis der boolesche Ausdruck den Wert „false“ zurückliefert.

## RESÜMEE: WHILE-Anweisung

- Kopfgesteuerte Schleife
- Anzahl der Durchläufe wird über Abbruchbedingung gesteuert (Steuervariable)
- Abbruchbedingung: boolescher Ausdruck ,
  (Abbruch bei „false“)

(Beispielaufgaben S. 69,70)

## REPEAT-Anweisung

- Der boolesche Ausdruck wird nach dem ersten Durchlaufen der Schleife geprüft.
  Es erfolgt wenigstens ein Schleifendurchlauf!!
  Diese Tatsache beeinflußt die Entscheidung bei der Wahl zwischen der WHILE-DO-Schleife und der REPEAT-UNTIL-Schleife.

- Wenn der Ausdruck den Wert „false“ zurückliefert, wird die Schleife erneut durchlaufen
- Der Vorgang wiederholt sich, bis der boolesche Ausdruck den Wert „true“ zurückliefert.
- Die Umklammerung mehrerer Anweisungen durch BEGIN...END ist bei REPEAT-UNTIL nicht notwendig, da REPEAT...UNTIL selbst als Umklammerung gilt.

## REPEAT-Anweisung:

- Fußgesteuerte Schleife
- Mindestens ein Durchlauf (!)
- Anzahl der Durchläufe wird über Abbruchbedingung gesteuert (Steuervariable)
- Abbruchbedingung: boolescher Ausdruck,
  (Abbruch bei „true“)

(Beispielaufgabe S. 71,72, 74)