

## COBOL Crash-Kurs

# COBOL = BCOOL

### Teil 1

**cps4it**

consulting, projektmanagement und seminare für die informationstechnologie

Ralf Seidler, Stromberger Straße 36A, 55411 Bingen

Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: [ralf.seidler@cps4it.de](mailto:ralf.seidler@cps4it.de)

Internet: <http://www.cps4it.de>

- 
- Seite 5: Einführung
  - Seite 16: Programmaufbau
  - Seite 48: Numerische Daten
  - Seite 74: Schleifen mit Zähler
  - Seite 80: Entscheidungen
  - Seite 106: Schleifen mit Bedingungen
  - Seite 111: Sections
  - Seite 124: Tabellenverarbeitung
  - Seite 131: sequentielle Dateien

- Sprache COBOL kennen lernen
- Praxisbeispiele kennen lernen
- üben ... üben ... üben – nur offline
- Besonderheiten

- 
- 
- A blue arrow pointing to the right, highlighting the first item in the list.
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien

Standard

IBM

Historie

CODASYL

ANSI

COBOL

## Entstehung

---

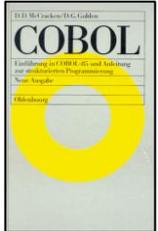
- Common Business Oriented Language
- überwiegend kaufmännischer Bereich
- leicht lesbar
- gut für das Bearbeiten von Daten
- Entwicklung auf Anregung von US-Verteidigungsministerium -> CODASYL
- Vorstellung Standard: 29. Mai 1959
- erster Standard April 1960 -> COBOL 60
- <https://www.it-times.de/news/google-baut-unterseekabel-grace-hopper-fuer-die-verbinding-der-usa-mit-europa-136080/>



- COBOL 60
- OS/VS COBOL (COBOL 68)
- VS COBOL II (ANS-85)
- COBOL/370
- COBOL for MVS and VM
- Enterprise COBOL for z/OS and OS/390
- zur Zeit:
  - Version 4 Release 2 – 30.04.2022 Support Ende
  - Version 6 Release 3
  - Version 6 Release 4

- Bookmanager
  - Programming Guide
  - Language Reference
  - local im Intranet oder im Internet bei IBM
  - [https://www.ibm.com/support/knowledgecenter/en/SS6SG3/entcobol\\_family\\_welcome.html](https://www.ibm.com/support/knowledgecenter/en/SS6SG3/entcobol_family_welcome.html)
- Internetseiten
  - siehe Suchmaschinen
  - [www.cobol-workshop.de](http://www.cobol-workshop.de)
  - COBOL in IBM-Comunities  
<https://community.ibm.com/community/user/ibmcommunity/search?executeSearch=true&SearchTerm=cobol&l=1>
  - etc.

- Oldenbourg Verlag
  - COBOL ~~€ 54,80~~
- Spektrum Akademie Verlag
  - Einführung in die Progr.sprache COBOL ~~€ 29,95~~
- siehe Suchmaschinen
  
- IBM Dokumentation  
<https://www-01.ibm.com/support/docview.wss?uid=swg27036733>
- Firmeninterne Unterlagen
  - Richtlinien



## Warum COBOL? – 1

---

- Mainframe Applikationen
  - sind groß, sehr groß, sehr sehr groß
  - haben sehr sehr komplexe Geschäftslogik
  - leben lange
  - sind dynamisch
  - sind geschäftskritisch
  - verarbeiten riesige Datenmengen
  - sind flexibel für neue Technik

## Warum COBOL? – 2

---

- COBOL ist
  - selbst-dokumentierend
  - einfach zu lernen
  - portierbar
  - performant
  - skalierbar
  - universell benutzt
  - offen
  - vollständig
  - erweiterbar
  - gut und einfach wartbar

## Warum COBOL? – 3

---

- Nehmen wir an, alle Firmen, die COBOL einsetzen, würden in eine andere Sprache und Plattform investieren:
  - Sie wären – nach seriösen Hochrechnungen – mindestens 10 Jahre alleine damit beschäftigt, die Anwendungen zu portieren.
  - Sie hätten Anwendungen, die langsamer laufen.
  - Sie hätten Anwendungen, die (wieder) fehlerhaft sind.
  - ...
- Das passiert nicht, so wie Englisch nicht abgeschafft wird. ;-))

## Warum COBOL? – 4

---

- COBOL wurde mehrfach tot gesagt durch
  - Fortran: 1960s
  - PL/I: 1970s
  - PASCAL: 1980s
  - Smalltalk: 1985
  - C: 1990
  - C++: 1995
  - Java: 1998
  - C#: 2001
  - Cloud: 2018 “Mainframe ist bald tot”

Lassen wir uns überraschen. ;-)

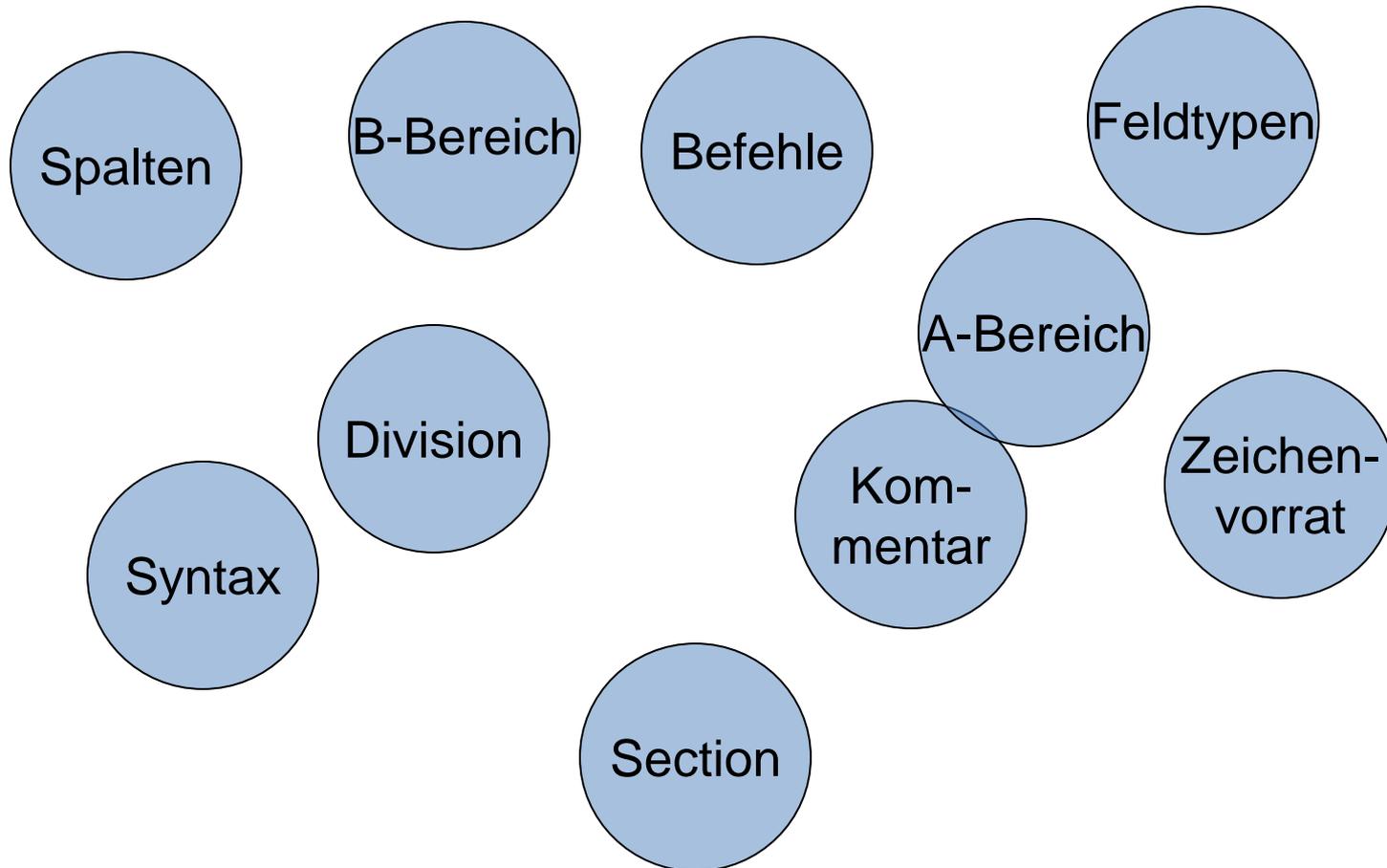
- Fazit:  
COBOL is the language of the future! ;-)



**COBOL = BCOOL**



- 
- Einführung
  - ➔ • Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien



# Programmaufbau

## ein einfaches Programm – 1

---

IDENTIFICATION DIVISION.

PROGRAM-ID.        XXZ01.

\*

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

        SELECT AUSGABE TO DRUCKER.

\*

DATA DIVISION.

FILE SECTION.

FD     AUSGABE.

\*

01     AUSGABE-SATZ            PIC X(80) .

\*

# Programmaufbau

## ein einfaches Programm – 2

---

\*

```
PROCEDURE DIVISION.
```

```
ENDLICH-PROGRAMM-CODE SECTION.
```

\*

```
OPEN OUTPUT AUSGABE.
```

\*

```
MOVE 'http://www.cps4it.de' TO AUSGABE-SATZ
```

```
WRITE AUSGABE-SATZ
```

\*

```
MOVE 'das ist eine web-Adresse' TO AUSGABE-SATZ
```

```
WRITE AUSGABE-SATZ
```

\*

```
CLOSE AUSGABE
```

```
GOBACK.
```

# Programmaufbau

## ein einfaches Programm – 3

---

IDENTIFICATION DIVISION.

PROGRAM-ID.        XXZ01.

\*

PROCEDURE DIVISION.

\*

      display 'hello world'

\*

      GOBACK.

- IDENTIFICATION DIVISION
  - Erkennungsteil – muss
- ENVIRONMENT DIVISION
  - Beschreibung der Umgebung
- DATA DIVISION
  - Datenteil
- PROCEDURE DIVISION
  - Prozedurteil

## Zeichenvorrat von COBOL – 1

---

- A-Z
  - COBOL-Wörter, Benutzerwörter (z.B. Variablen)
- a-z
  - COBOL-Wörter, Benutzerwörter
- 0-9
  - Benutzerwörter, numerische Operationen, numerische Konstanten
- blank
  - Trennzeichen

## Zeichenvorrat von COBOL – 2

---

- \$ €
  - Währungszeichen für Druckaufbereitung
- ,
  - Druckaufbereitung, Trennung von Wörtern, Trennung von Indizes bei Tabellen
- .
  - Druckaufbereitung, Abschluss von Anweisungen, Abschluss von Überschriften

## Zeichenvorrat von COBOL – 3

---

- ;
  - Trennung von Anweisungen
- =
  - Wertzuweisung, Vergleichsoperator
- -
  - Bildung von Wörtern, numerischer Operator, Zeichen für Druckaufbereitung
- +
  - numerischer Operator, Druckaufbereitung

## Zeichenvorrat von COBOL – 4

---

- ( )
  - algebraische Klammerung, Tabellenelemente
- \*
  - Druckaufbereitung, Multiplikation
- \*\*
  - Exponentiation
- /
  - Bruchstrich, Seitenvorschub

## Zeichenvorrat von COBOL – 5

---

- > < >= <=
  - Bedingungen
- ‘ “
  - Begrenzungszeichen für nicht numerische Literale;  
ANSI: “
- :
  - Teilzeichenketten
- \*>
  - Inline-Kommentar



## Wörter und Literale – 1

---

- Wörter sind
  - reservierte COBOL-Wörter
  - Benutzerwörter
- Regeln
  - Folge von max. 30 Zeichen
  - erlaubt sind Buchstaben, Ziffern und -
  - mindestens 1 Buchstabe
  - erstes und letztes Zeichen kein -
  - Namen für Datenfelder müssen eindeutig sein

## Wörter und Literale – 2

---

- sinnvoll
  - Namen sprechend wählen
  - lange Namen mit Bindestrichen übersichtlich teilen
  - sinnvolle Abkürzungen wählen
- Beispiele:
  - SUMME
  - A3BX45
  - ~~SUMMEDERVERKAUFSNDLSOST~~
  - MENGE-EIN
  - ~~GRÖSSER~~



## Figurative Konstanten

---

- [ALL] ZERO / ZEROS / ZEROES
- [ALL] SPACE / SPACES
- [ALL] HIGH-VALUE / HIGH-VALUES
- [ALL] LOW-VALUE / LOW-VALUES
- [ALL] QUOTE / QUOTES
- ALL 'literal'
- [ALL] NULL / NULLS



## Aufteilung der Spalten – 1

---

- 01-06 Zeilennummerierung möglich
- 07 Typ der Zeile
  - blank normale Zeile
  - \* Kommentarzeile
  - - Fortsetzungszeile
  - / Seitenvorschub für Druck

## Aufteilung der Spalten – 2

---

- 08-11 A-Bereich
  - Überschriften von DIVISIONs, SECTIONs
  - bestimmte Stufennummern (später)
- 12-72 B-Bereich
  - Namen der Felddefinitionen
  - Befehle
- 73-80 beliebig, unabhängig von COBOL

(ISPF-Profil: NUM COB STD  
-> bitte nicht mehr benutzen)



## IDENTIFICATION DIVISION – 1

---

- Inhalte
  - IDENTIFICATION DIVISION.
  - PROGRAM-ID.            Programmname.
  - [AUTHOR.                Verfasser.]
  - [INSTALLATION.        Computertyp.]
  - [DATE-WRITTEN.        Datum der Erstellung.]
  - [DATE-COMPILED.      Datum der Umwandlung.]
  - [SECURITY.             Wer darf Programm lesen.]
- 1. Teil im A-Bereich, Inhalte im B-Bereich

- Position
  - Überschriften / Paragraphen im A-Bereich
  - Inhalte im B-Bereich
- Punkte bei Paragraphen und Inhalten wahlfrei
- Reihenfolge muss nach Norm eingehalten werden; Compiler braucht das nicht
- dringende Empfehlung: nur Mussangaben verwenden, da die Kannangaben von ANSI als “obsolete” gekennzeichnet wurden



- Beschreibung der Umgebung
  - Namen (special-names)
  - Organisationsformen
  - Zugriffsmethoden
- 2 Kapitel
  - CONFIGURATION SECTION
  - INPUT-OUTPUT SECTION

- Inhalte
  - [ENVIRONMENT DIVISION.]
  - [CONFIGURATION SECTION.]
  - [SOURCE-COMPUTER.      Comp-name-1  
                                  ~~[with debugging mode]].~~
  - [OBJECT-COMPUTER.      Comp-name-2.]
  - [SPECIAL-NAMES.        Spezialangaben.]
- 1. Teil im A-Bereich, Inhalte im B-Bereich

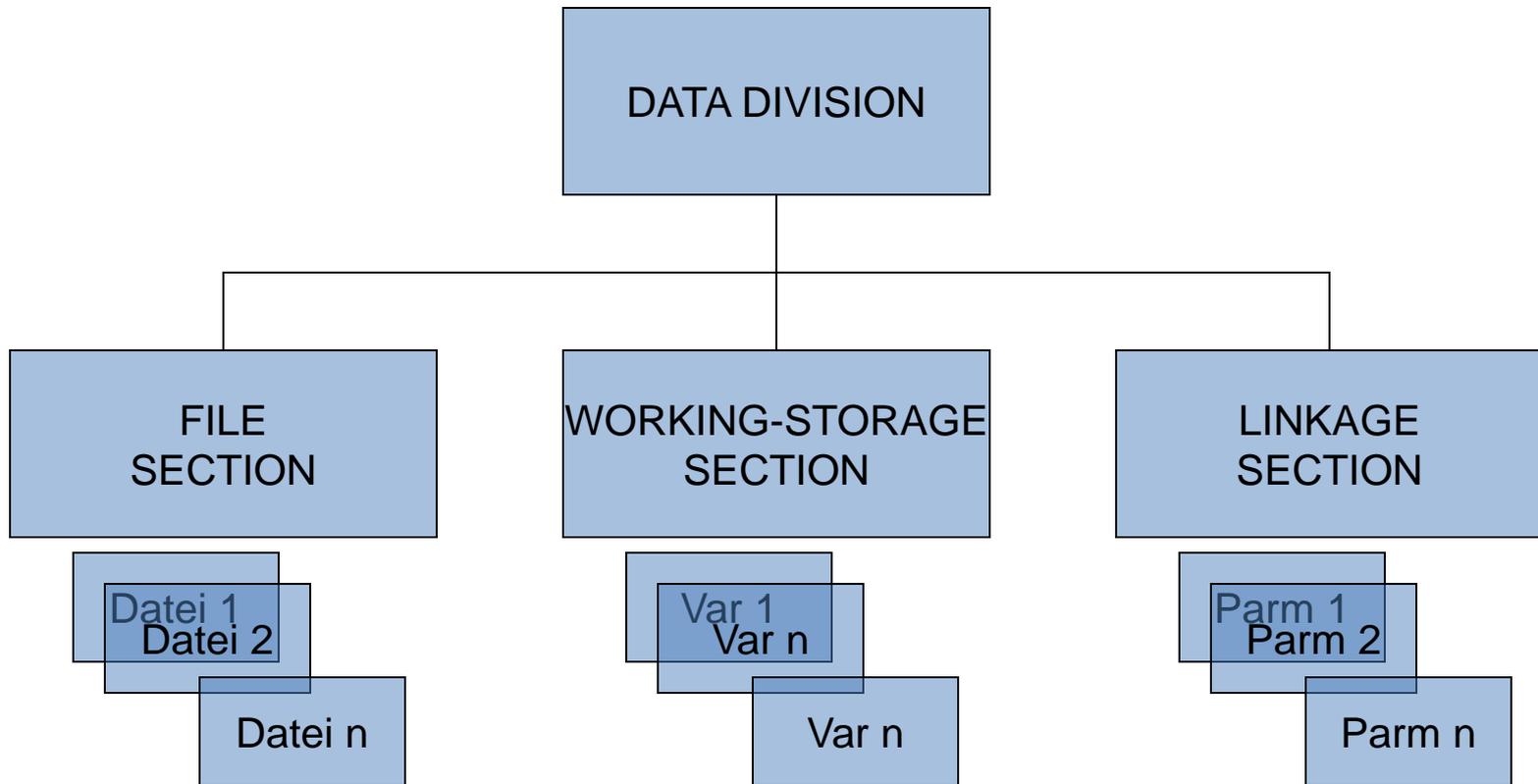
- Beispiele für SPECIAL-NAMES
  - DECIMAL-POINT IS COMMA.
  - SYSOUT IS AUSGABE.
  - PAGE IS NEUE-SEITE.
  - ALPHA-TEST IS 'A' ALSO 'a' etc.
- Ergebnisse
  - Komma wird benutzt
  - DISPLAY UPON AUSGABE. (statt SYSOUT)
  - WRITE ... AFTER ... NEUE-SEITE (statt PAGE)
  - SORT mit a = A / auch Abfragen? -> testen!

- Organisationsform der Dateien
  - direkt, relativ, sequentiell
- Zugriffsmethode
  - direkt, sequentiell, dynamisch
  - Returncode Behandlung
- Beispiele später



## DATA DIVISION

---



## DATA DIVISION – Datendefinition

---

- Format:
- Stufennummer                      datenname                      KLAUSEL

- Beispiel:

\*

```
01  EINGABE-SATZ          PIC  X(80) .
01  AUSGABE-SATZ         PIC  X(80) .
01  ZIFFER                PIC  9(01) .
01  FUENF-ZIFFERN        PIC  9(05) .
```

789012345

## Feldtypen

---

- alphanummerisch
  - 01 FELD PIC **X**(020).
- alphabetisch
  - 01 FELD PIC **A**(020).
- numerisch binär
  - 01 FELD-BINAER PIC [S]**9**(08) **BINARY**.
- numerisch dezimal
  - 01 FELD-DECIMAL PIC [S]**9**(5)V99  
**PACKED-DECIMAL**.
- numerisch Display-Feld
  - 01 FELD-DISPLAY PIC [S]**9**(08).

## Darstellung im hexadezimalen Format

---

- alphanummerisch / alphabetisch
  - ABEND:            C1 C2 C5 D5 C4
- numerisch binär
  - 12345:            00 00 30 39
- numerisch dezimal
  - 12345:            12 34 5F                            12 34 5C
- numerisch Display-Feld
  - 12345:            F1 F2 F3 F4 F5                    F1 F2 F3 F4 C5

## Stufennummern

---

- 77 war früher “das übliche” (heute 01 nehmen)
- 66 für Redefinitionen (nicht empfohlen)
- 01, 05, ... für Strukturen
  - Beginn auf Doppelwortgrenze ausgerichtet
- 88 Schalter

- Format:

Stufennummer { datenname  
FILLER  
blank } KLAUSEL

## DATA DIVISION – Datendefinition – Struktur

```
*  
01 Customer-Record.  
    05 Customer-Name.  
        10 Last-Name          Pic x(17) .  
        10 [Filler]           Pic x .  
        10 Initials           Pic xx .  
    05 Part-Order.  
        10 Part-Name          Pic x(15) .  
        10 Part-Color         Pic x(10) .
```

01 und 77: A-Bereich  
02 bis 49: B-Bereich

```
000111111  
789012345
```



- Die Logik des Programms oder der Code
- Aufteilung nach Regeln der strukturierten Programmierung möglich und dringend zu empfehlen - > später



## einfache Befehle – 1

---

- **MOVE**
  - alpha to alpha
  - alpha to numeric
  - numeric to alpha
  - numeric to numeric
- **DISPLAY**
  - Anzeige von Daten - normalerweise Testhilfe!
  - Schreiben auf SYSOUT
- **ACCEPT**
  - Lesen vom System, von SYSIN

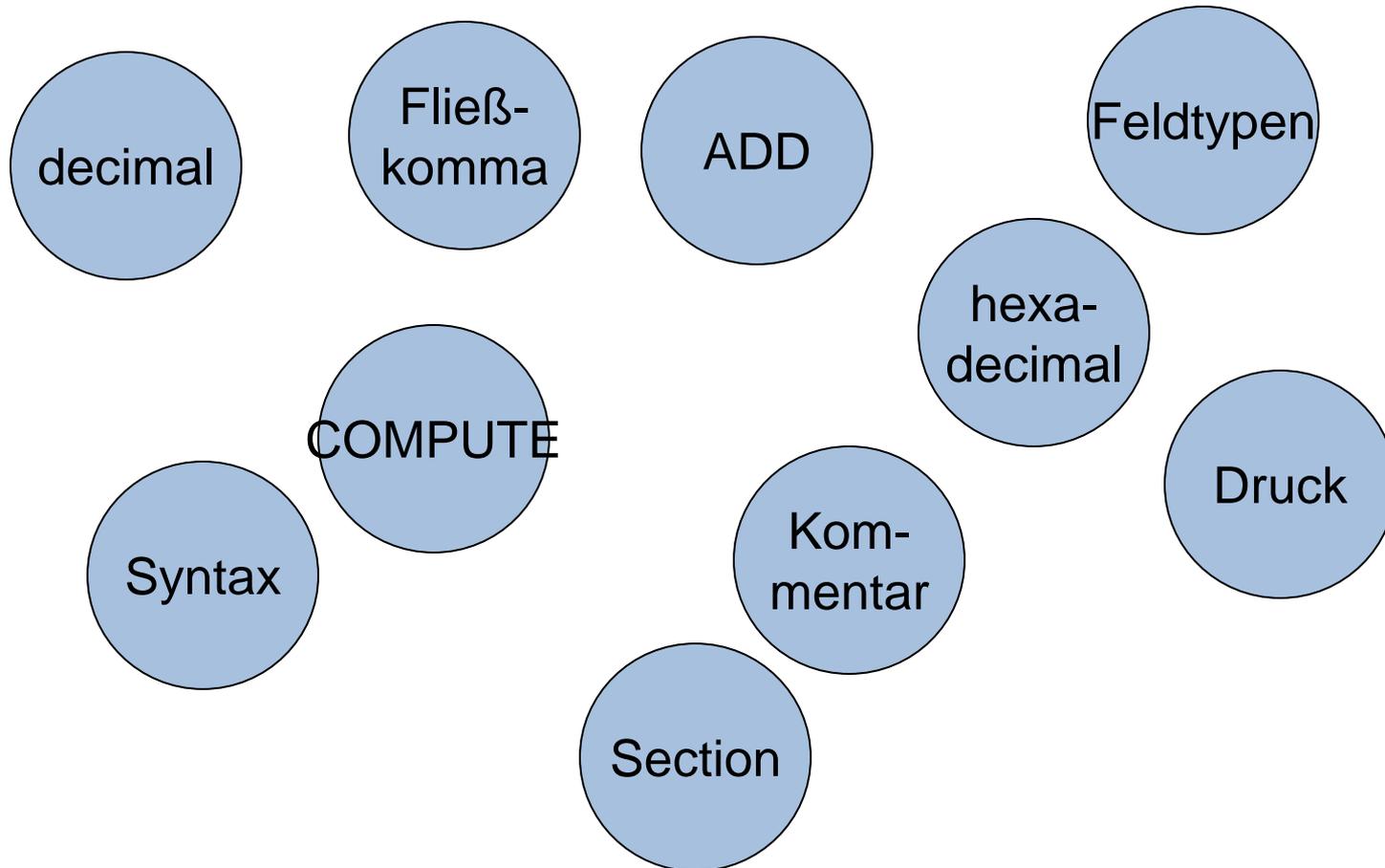
- CONTINUE
  - Leeranweisung
  - sehr zu empfehlen
  - sinnvoll mit einem Punkt, um das Ende einer Verarbeitung zu kennzeichnen
  - sinnvoll innerhalb von Bedingungen



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - • Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien

## Begriffe

---



## External decimal

---

- PIC S9999 [DISPLAY]

  - +1234            F1 F2 F3 C4

  - -1234            F1 F2 F3 D4

  - 1234             F1 F2 F3 C4

- PIC 9999 [DISPLAY]

  - 1234             F1 F2 F3 F4

Auch mit Parameter  
leading sign möglich

\*

```
01  WERT-OHNE-VZ      PIC  9999 .
77  WERT-MIT-VZ       PIC  S9(4) .
01  WERT-MIT-VZ       PIC  S9(04) DISPLAY .
```

## Internal decimal

---

- PIC S9(5) PACKED-DECIMAL oder
- PIC S9(5) COMP-3
  - +1234            01 23 4C
  - -1234            01 23 4D
- PIC 9(5) PACKED-DECIMAL oder COMP-3
  - +1234            01 23 4F
  - -1234            01 23 4F

\*

```
01  WERT-MIT-VZ            PIC S99999 COMP-3.
```

## binary

---

- PIC S9(4) BINARY / COMP / COMP-4 / COMP-5
  - +1234            04 D2
  - -1234            FB 2E
- PIC 9(4) BINARY / COMP / COMP-4 / COMP-5
  - +1234            04 D2

\*

```
01  WERT-MIT-VZ            PIC S9999 COMP.
```

## Internal Floating Point

---

- COMP-1

  - +1234            43 4D 20 00

- COMP-2

  - +1234            43 4D 20 00 00 00 00 00

  - -1234            C3 4D 20 00 00 00 00 00

\*

```
01  WERT-MIT-FP      COMP-1 .
```

- Logik

  - The leftmost bit contains the sign and the next 7 bits contain the exponent; the remaining 3 or 7 bytes contain the mantissa.

- Hinweis: benutzt eigene Register; seit 64-bit-Architektur sehr schnell!

## External Floating Point

---

- PIC +9(2).9(2)E+99 [DISPLAY]
  - +1234                    4E F1 F2 4B F3 F4 C5 4E F0 F2
  - -1234                    60 F1 F2 4B F3 F4 C5 4E F0 F2

\*

```
01  WERT-MIT-EXP            PIC +99.99E+99.
```

## Schreibweise und sinnvolle Definitionen

---

- BINARY (COMP / COMP-4 / COMP-5)
- PACKED-DECIMAL (COMP-3)
  
- immer mit Vorzeichen (schneller)
- gepackt immer mit ungerader Zifferanzahl
- binär immer S9(4) bzw. S9(8)
- Stellen > 9: nutze gepackt statt binär (noch)

## Rechenbefehle

---

- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- COMPUTE

## Rechenbefehl ADD

---

- Syntax in Kurzfassung und Beispielen

```
ADD W1 TO W2
```

```
ADD 12 TO W2
```

```
ADD 12 TO W2 ROUNDED
```

```
ADD 12 TO 45 GIVING W3 ROUNDED
```

```
ADD CORR STR1 TO STR2 ROUNDED
```

```
END-ADD
```

- Details siehe Language Reference

## Rechenbefehl SUBTRACT

---

- Syntax in Kurzfassung und Beispielen

```
SUBTRACT W1 FROM W2
```

```
SUBTRACT 12 FROM W2
```

```
SUBTRACT 12 FROM W2 ROUNDED
```

```
SUBTRACT 12 FROM 45 GIVING W3 ROUNDED
```

```
SUBTRACT CORR STR1 FROM STR2 ROUNDED
```

```
END-SUBTRACT
```

- Details siehe Language Reference

## Rechenbefehl MULTIPLY

---

- Syntax in Kurzfassung und Beispielen

```
MULTIPLY W1 BY W2
```

```
MULTIPLY 12 BY W2
```

```
MULTIPLY 12 BY W2 ROUNDED
```

```
MULTIPLY 12 BY 45 GIVING W3 ROUNDED
```

```
MULTIPLY CORR STR1 BY STR2 ROUNDED
```

```
END-MULTIPLY
```

- Details siehe Language Reference

## Rechenbefehl DIVIDE

---

- Syntax in Kurzfassung und Beispielen

```
DIVIDE W1 INTO W2
```

```
DIVIDE 12 INTO W2
```

```
DIVIDE 12 INTO W2 ROUNDED
```

```
DIVIDE 12 INTO 45 GIVING W3 ROUNDED
```

```
DIVIDE CORR STR1 INTO STR2 ROUNDED
```

```
END-DIVIDE
```

- Details siehe Language Reference

## Rechenbefehl COMPUTE

---

- Syntax in Kurzfassung und Beispielen

```
COMPUTE W1 = W1 + W2
```

```
COMPUTE W1 ROUNDED = W1 + W2
```

```
COMPUTE W1 = W1**W2 + W2/4 - 2003
```

```
ON SIZE ERROR mach was
```

```
NOT ON SIZE ERROR mach was anderes
```

```
END-COMPUTE
```

- Details siehe Language Reference



## Druckaufbereitung – Nullenunterdrückung

---

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
    05  FELD1  PIC  99999.  
*      12345      12345  
    05  FELD2  PIC  ZZZ99.  
*      00345      bb345  
*      00005      bbb05  
    05  FELD3  PIC  ZZZZZ.  
*      00000      bbbbb.  
*      00345      bb345
```

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
    05  FELD1  PIC  €99999.  
*      12345      €12345  
    05  FELD2  PIC  €ZZZ99.  
*      00345      €bb345  
*      00005      €bbb05  
    05  FELD3  PIC  €ZZZZZ.  
*      00000      €bbbbb.  
*      00345      €bb345
```

SPECIAL-NAMES:  
CURRENCY SIGN IS "€".

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
05  FELD1  PIC  €€€999.  
*      12345      €12345  
05  FELD2  PIC  €€€€€9.  
*      00345      bb€345  
*      00005      bbbb€5  
05  FELD3  PIC  €€€€€€.  
*      00000      bbbbbb.  
*      00345      bb€345
```

SPECIAL-NAMES:  
CURRENCY SIGN IS "€".

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
05  FELD1  PIC  999,999.  
*      123456      123,456  
05  FELD2  PIC  ZZZ,ZZZ.  
*      003456      bb3,456  
*      000056      bbbbb56  
05  FELD3  PIC  ZZ,9999.  
*      000000      bbb0000.
```

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
05  FELD1  PIC  999,999.  
*      123,456    123,456  
*      000,025    000,025  
05  FELD2  PIC  ZZZ,ZZZ.  
*      000,025    bbbbb25  
05  FELD3  PIC  ZZ9,999.  
*      000,025    bb0,025
```

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

01 FOLDER-FUER-DRUCK.

05 FELD1 PIC 99,99.

\* 123,456 12,34

05 FELD2 PIC ZZZZZ9,9.

\* 000,056 bbbbbb0,0

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
05  FELD1  PIC  999999- .  
*      -123456      123456-  
05  FELD2  PIC  999999+ .  
*      -123456      123456-  
*      +000056      000056+
```

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
    05  FELD1  PIC  +ZZZZZ9.  
*      +000056      +bbbb56  
    05  FELD2  PIC  ++++++9.  
*      +000056      bbbb+56  
    05  FELD3  PIC  -----99.  
*      +003456      bbb3456
```

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

01 FOLDER-FUER-DRUCK.

05 FELD1 PIC 999999DB.

\* -123456 123456DB

\* +000056 000056bb

05 FELD2 PIC PIC 999999CR.

\* -123456 123456CR

\* +000056 000056bb

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.  
    05  FELD1  PIC  ****99.  
*      123456      123456  
*      +000056     ****56  
    05  FELD2  PIC  *****.  
*      000056     ****56  
*      000000     *****  
    05  FELD3  PIC  ***999.  
*      000056     ***056
```

SPECIAL-NAMES:  
DECIMAL-POINT  
IS COMMA.

- Syntaxbeispiele

```
01  FELDER-FUER-DRUCK.
```

```
05  FELD1 PIC 999B999.
```

```
*      123456      123 456
```

```
*      +000056     000 056
```

```
05  FELD2 PIC 99B99B99.
```

```
*      123456      12 34 56
```

```
05  FELD3 PIC 099B99B99.
```

```
*      123456      012 34 56
```

- Syntaxbeispiele

```
01    FELDER-FUER-DRUCK.
```

```
05    FELD1  PIC  XBXBXB.
```

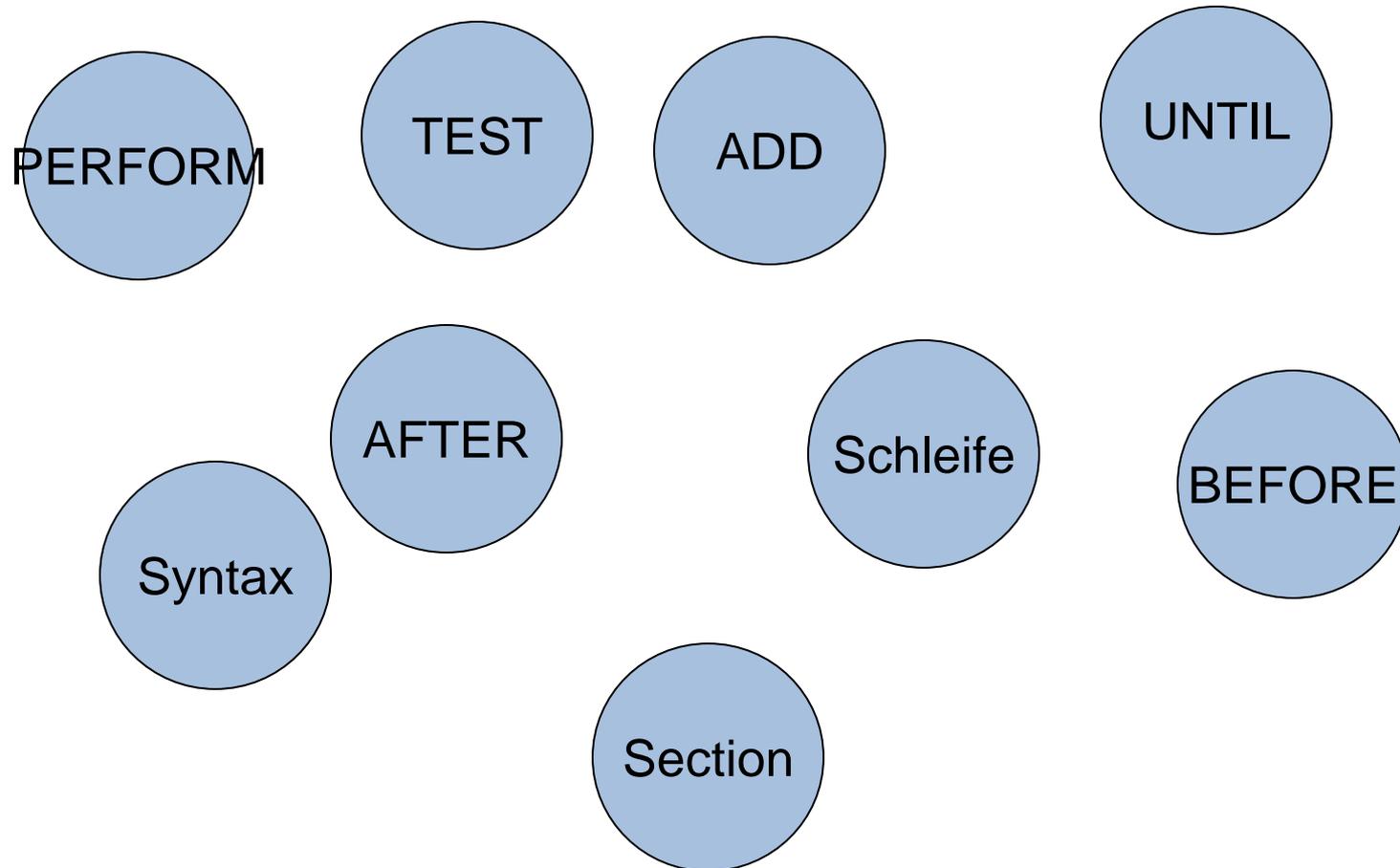
```
*           COBOL      C O B O L
```

```
05    FELD2  PIC  99/99/99.
```

```
*           100809     10/08/09
```



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - • Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien



# Schleifen mit Zähler

## einfache Schleife – DO UNTIL

- Syntaxbeispiel

**Anweisung-1**

**PERFORM WITH TEST AFTER**

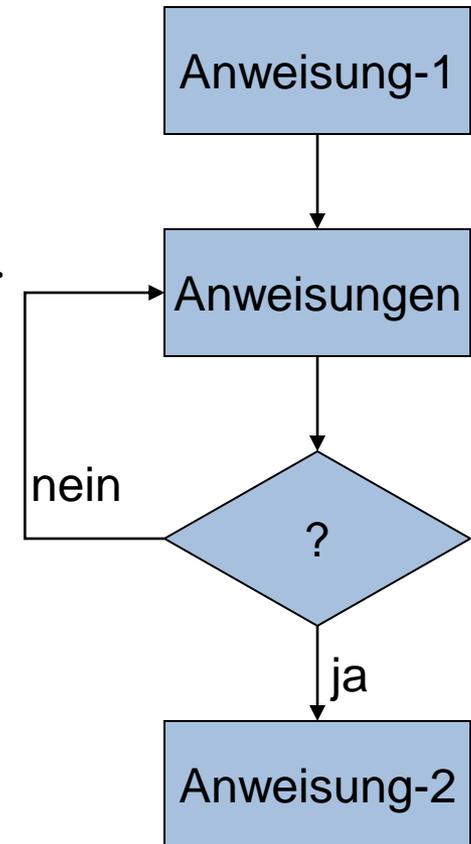
**VARYING I1 FROM 1 BY 1**

**UNTIL I1 > 10**

**anweisung(en)**

**END-PERFORM**

**Anweisung-2**



# Schleifen mit Zähler

## einfache Schleife – DO WHILE

- Syntaxbeispiel

Anweisung-1

```
PERFORM [WITH TEST BEFORE]
```

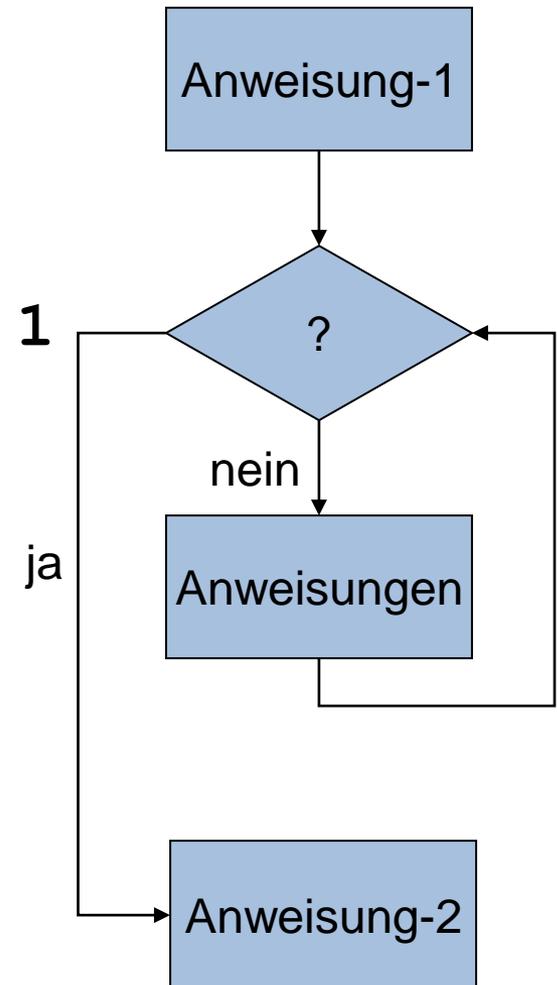
```
    VARYING I1 FROM 1 BY 1
```

```
    UNTIL I1 > 10
```

```
    anweisung(en)
```

```
END-PERFORM
```

Anweisung-2

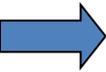


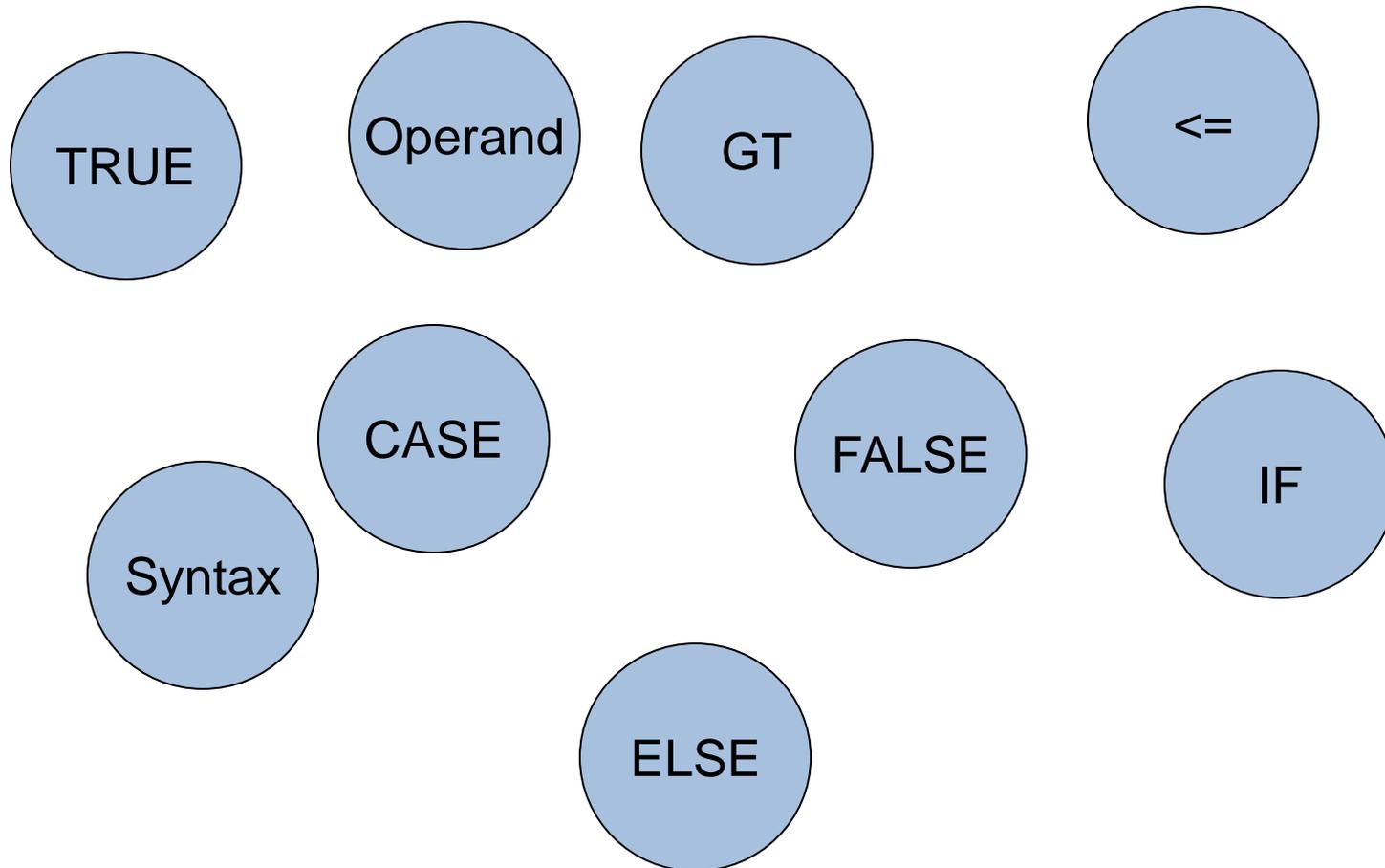
- Syntaxbeispiel

```
PERFORM [WITH TEST {BEFORE|AFTER}]  
        VARYING var-1 FROM {var-2|lit-2}  
                BY {var-3|lit-3}  
        UNTIL  Bedingung  
        anweisung(en)  
END-PERFORM
```

- CONTINUE
  - Leeranweisung
  - sehr zu empfehlen
  - sinnvoll mit einem Punkt genau dann, um das Ende einer Verarbeitung zu kennzeichnen
  - sinnvoll innerhalb von Bedingungen (ohne Punkt)



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  -  • Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien



## Alternative

---

- Alternative
  - wahr oder falsch (TRUE oder FALSE)
- Prüfung durch Bedingungen
  - Vergleichsbedingung
  - Vorzeichenbedingung
  - Klassenbedingung
  - Bedingungsnamenbedingungen

## Beispiel

---

```
IF    A > B
THEN
    IF    B > C
    THEN
        CONTINUE
    ELSE
        tu was-1
    END-IF
    tu was-2
ELSE
    tu was-3
END-IF
```

## Syntax

---

**IF**     Bedingung

**[THEN]**

        {anw-1-1 [anw-1-2] | CONTINUE} ...

**[ELSE**

        {anw-2-1 [anw-2-2] | CONTINUE} ...]

**[END-IF]**

## Vergleichsbedingung

---

- Vergleich eines Subjekts und einem Objekt mit einem Vergleichsoperator
  - subject [IS] [NOT] oper object
  - subject ist Datenfeld oder Literal oder arithmetischer Ausdruck
  - object ist Datenfeld oder Literal oder arithmetischer Ausdruck
  - oper ist
    - GREATER THAN | LESS THAN | EQUAL TO
    - > < = >= <=

## Vergleichsbedingung – Beispiele

---

**IF RECHENFELD > 100**

**IF SUMMENFELD = ANZAHL \* PREIS-PRO-STUECK  
- (ANZAHL / 1000 \* SKONTO)**

**IF BESTAND < MINDESTBESTAND**

**IF LIEFERDATUM > VERFALLDATUM**

## Vergleichsbedingung – numerische Daten – 1

---

- Vergleich eines Subjekts und einem Objekt mit einem Vergleichsoperator
  - subject [IS] [NOT] oper object
  - subject ist Datenfeld oder Literal oder arithmetischer Ausdruck
  - object ist Datenfeld oder Literal oder arithmetischer Ausdruck
  - oper ist
    - GREATER THAN | LESS THAN | EQUAL TO
    - > < = >= <=

## Vergleichsbedingung – numerische Daten – 2

---

- Vergleich auf Inhalt der Felder
- Vergleich nicht auf Länge der Felder
- Existenz von Kommata
- Existenz von Vorzeichen
- unterschiedliche Formate werden konvertiert

## Vergleichsbedingung – nicht-nummerische Daten

---

- Gültige Sortierfolge wird berücksichtigt
  - siehe SPECIAL-NAMES. ALPHA-TEST IS ...
- Vergleich Zeichen für Zeichen ab links
  - Felder gleicher Länge
    - bis Ergebnis
  - Felder ungleicher Länge (aufpassen!!)
    - bis Ergebnis
    - bis Ende kürzeres Feld ist dann der Rest des längeren Feldes blank, sind diese Felder gleich, sonst ist längeres Feld größer

## Vergleichsbedingung – nicht-nummerisch / numerisch gemischt

---

- Es gilt nicht der arithmetische Wert des numerischen Feldes
- Intensive Kenntnisse der internen Darstellung von Zahlen ist erforderlich
- Bei Benutzung wird gewarnt durch Compiler
- zurückhaltende Benutzung dringend empfohlen

## Vorzeichenbedingung

---

- Prüfung auf kleiner / gleich / größer Null
- Format
  - var-1                    [IS] [NOT] {POSITIVE | ZERO | NEGATIVE
  - arithm-1                [IS] [NOT] {POSITIVE | ZERO | NEGATIVE

## Klassenbedingung

---

- Prüfung auf
  - numerisch
  - alfabetisch
  - alfabetisch in Großbuchstaben
  - alfabetisch in Kleinbuchstaben
- Format
  - var-1    [IS] [NOT]    {NUMERIC | ALPHABETIC |  
ALPHABETIC-UPPER |  
ALPHABETIC-LOWER}

## Klassenbedingung – gültige Formen

---

Art des Kennzeichners	gültige Form der Klassenprüfung	
alfabetisch	ALPHABETIC	NOT ALPHABETIC
alfanummerisch	ALPHABETIC NUMERIC	NOT ALPHABETIC NOT NUMERIC
numerisch (COMP ...)	NUMERIC	NOT NUMERIC

## Klassenbedingung – FALSE als Ergebnis

---

- Bei Vergleichen ergibt sich der Wert FALSE bei
  - NUMERIC, wenn Inhalt nicht numerisch
  - ALPHABETIC, wenn andere Zeichen als Buchstaben oder Blank
  - ALPHABETIC-UPPER, wenn andere Zeichen als Großbuchstaben oder Blank
  - ALPHABETIC-LOWER, wenn andere Zeichen als Kleinbuchstaben oder Blank

## Bedingungsnamenbedingung

---

- Stufennummer 88
- Beispiel:

```
01  EINGABE-SATZ .  
    05  SATZART          PIC  X(01) .  
        88  SATZART-OK   VALUE  '1' THRU  '4' , '7' .  
    05  FAMILIENSTAND  PIC  9(01) .  
        88  LEDIG        VALUE  1 .  
        88  VERHEIRATET  VALUE  2 .  
        88  GESCHIEDEN   VALUE  3 .  
        88  SONSTIGES    VALUE  4 .
```

## Bedingungsnamenbedingung - Vorteile

---

- Der Vergleich ist schnell (naja 😞)
- Aussagefähigkeit d.h. leichter lesbar
- änderungsfreundlich
- kürzerer Code

- Bedingungen miteinander verknüpfen
  - AND
  - OR
  - Klammernregel beachten

A1	A2	A1 OR A2	A1 AND A2
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F

## geschachtelte IF-Anweisungen

---

- Innerhalb der Zweige können weitere IF-Anweisungen kodiert werden
- für Lesbarkeit
  - einrücken
  - END-IF benutzen
- CONTINUE benutzen
- keinen Punkt benutzen!

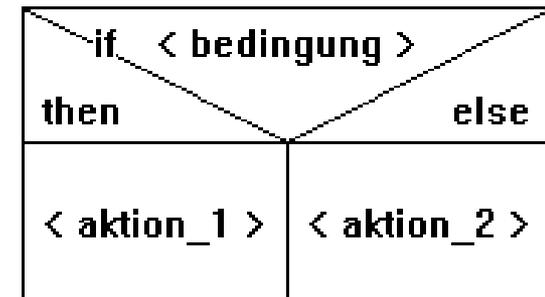
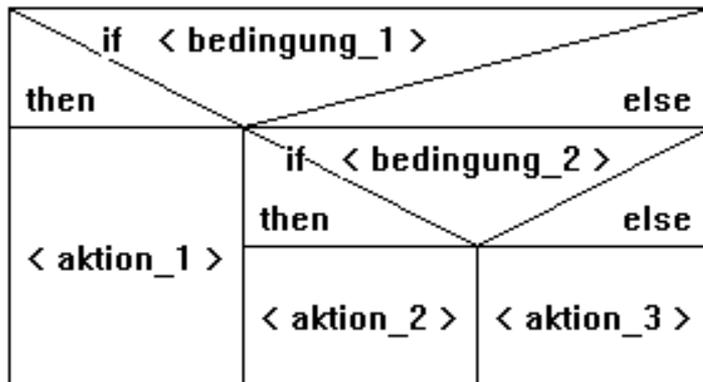
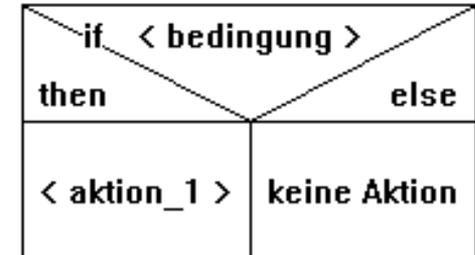
## geschachtelte IF-Anweisungen – Regeln

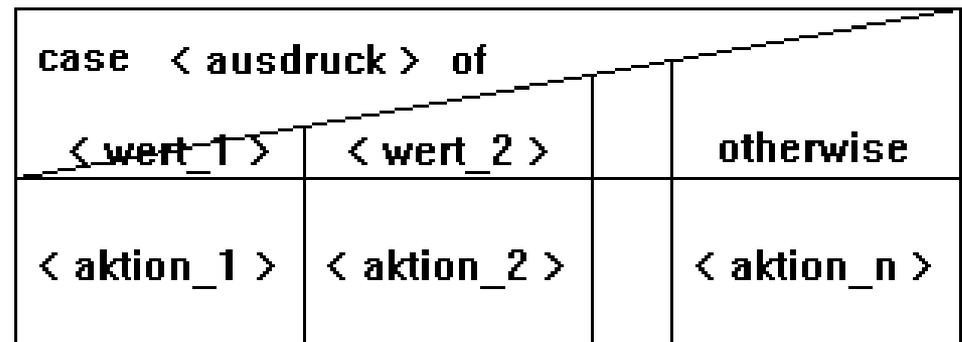
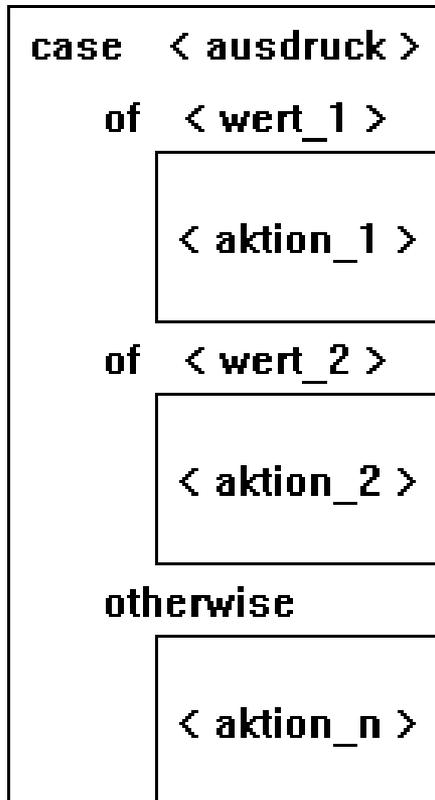
---

- Jede Anweisung hängt vom letzten IF oder ELSE ab
- Jedes ELSE innerhalb einer geschachtelten IF-Anweisung wird dem letzten IF zugeordnet, das noch nicht durch ein ELSE abgeschlossen worden ist.
- Vorsicht bei ELSE-Zweigen, die keine Aktion beinhalten. CONTINUE verwenden sinnvoll.

## Verzweigungen – Struktogramme

- Unvollständige Verzweigung
- vollständige Verzweigung
- Mehrfachverzweigung





- EVALUATE TRUE
- WHEN bedingung-1
  - anw-11 [... anw-1n]
- WHEN bedingung-2
  - anw-21 [... anw-2n]
- ...
- WHEN OTHER
  - anw-91 [... anw-9n]
- END-EVALUATE

- EVALUATE bedingung-1 ALSO bedingung-2
- WHEN TRUE ALSO TRUE
  - anw-11 [... anw-1n]
- WHEN FALSE ALSO TRUE
  - anw-21 [... anw-2n]
- ...
- WHEN OTHER
  - anw-91 [... anw-9n]
- END-EVALUATE

- EVALUATE TRUE ALSO FALSE
- WHEN bedingung-1 ALSO bedingung-2
  - anw-11 [... anw-1n]
- WHEN bedingung-1 ALSO bedingung-2
  - anw-21 [... anw-2n]
- ...
- WHEN OTHER
  - anw-91 [... anw-9n]
- END-EVALUATE

... und so weiter!  
Details siehe Bookmanager

Kodiere aber so, dass es  
\*jeder\* verstehen kann!

- Arithmetische Ausdrücke errechnen
- 1. WHEN-CLAUSE von links nach rechts
  - positiv, dann führe aus und springe zum Ende
- nächste WHEN-CLAUSE
  - positiv, dann führe aus und springe zum Ende
- etc. bis WHEN OTHER falls vorhanden
- für Performance: häufigsten Fall zu Beginn kodieren

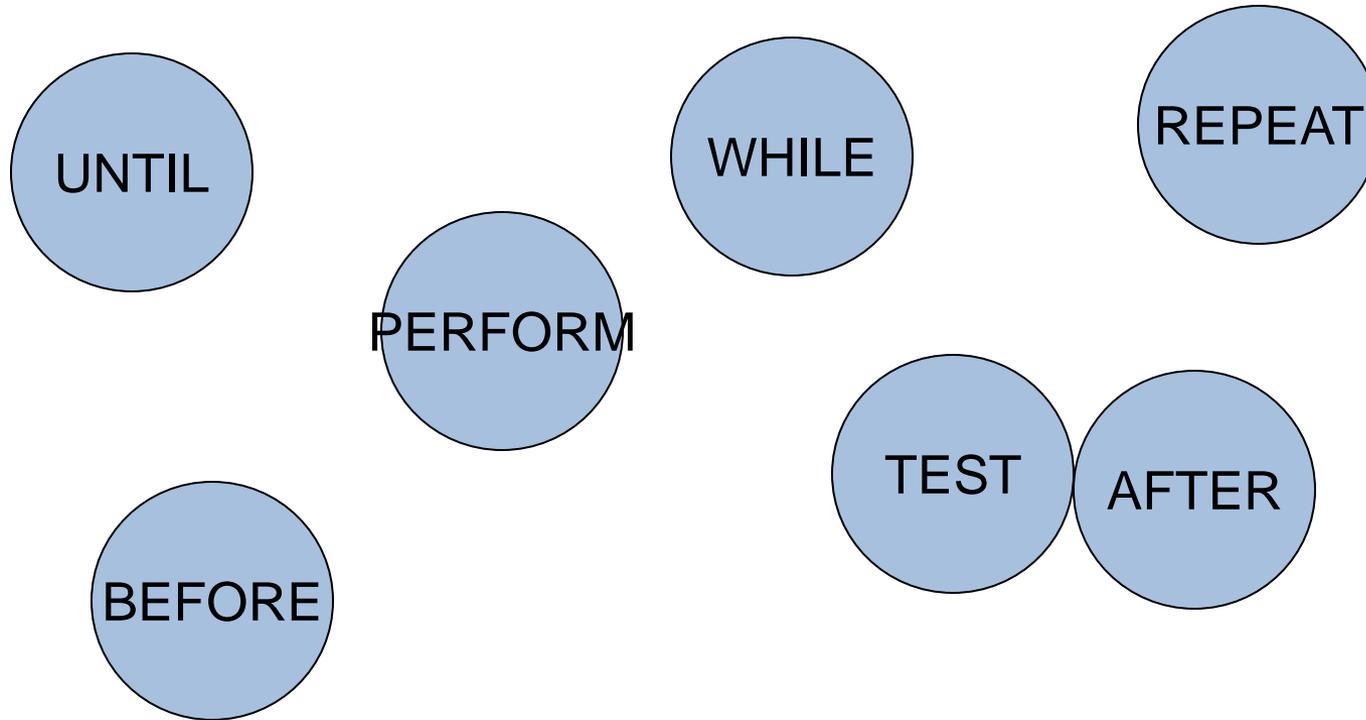


- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  -  • Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien

# Schleifen mit Bedingungen

## Begriffe

---

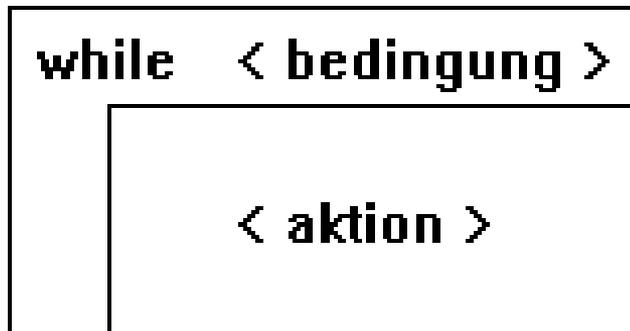


# Schleifen mit Bedingungen

## Abweisschleife (WHILE-Schleife)

---

- Die Aktion wird solange wiederholt, wie die Bedingung erfüllt ist.
- Die Bedingung wird vor der Aktion geprüft, d.h. die Aktion wird möglicherweise nie ausgeführt.



# Schleifen mit Bedingungen

## Nichtabweisschleife (UNTIL-Schleife)

---

- aktion wird solange wiederholt, bis die Bedingung erfüllt ist.
- Die Aktion wird immer mindestens einmal ausgeführt.

<b>repeat</b>	<b>&lt; aktion &gt;</b>
<b>until</b>	<b>&lt; bedingung &gt;</b>

# Schleifen mit Bedingungen

## Umsetzung in COBOL

---

- Abweisschleife

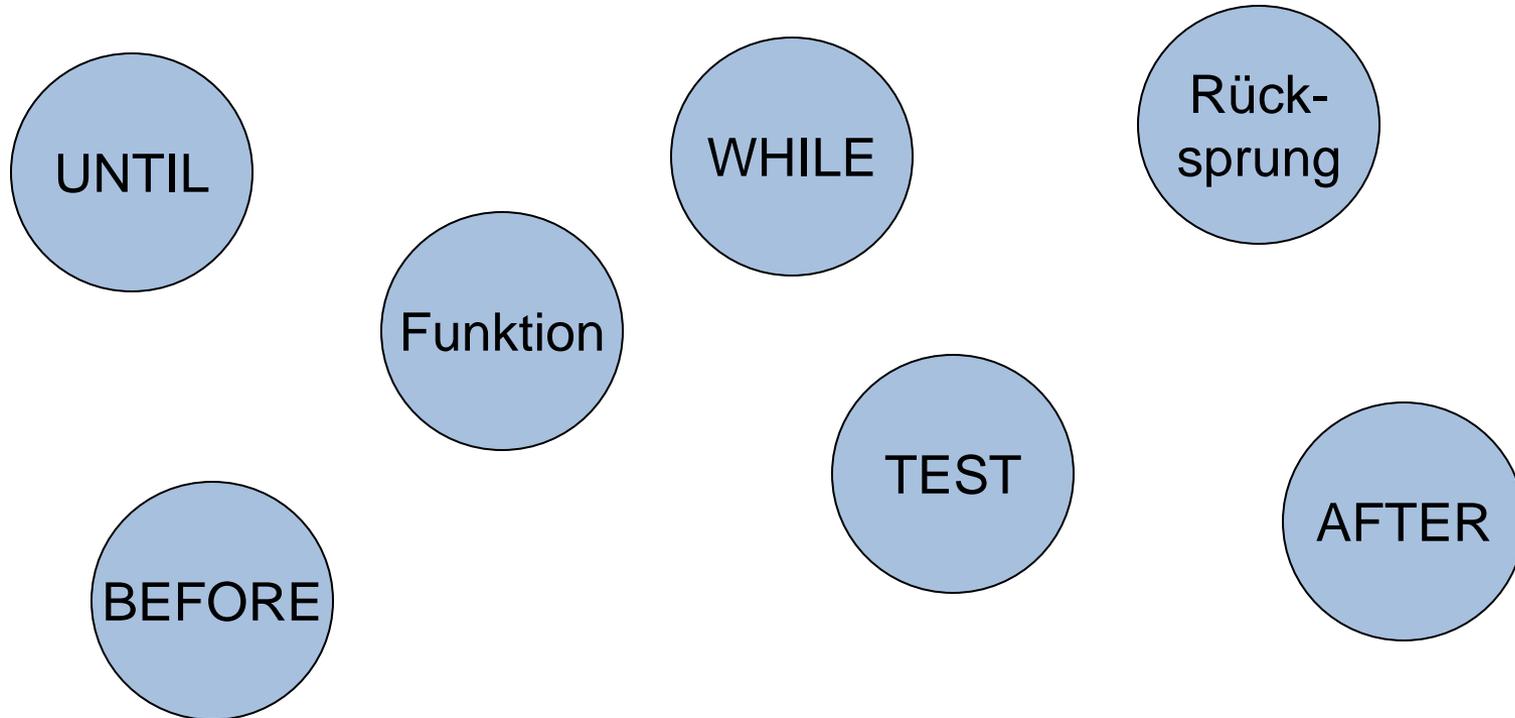
```
PERFORM [WITH TEST BEFORE]
        UNTIL   Bedingung
              anweisung(en)
END-PERFORM
```

- Nichtabweisschleife

```
PERFORM WITH TEST AFTER
        UNTIL   Bedingung
              anweisung(en)
END-PERFORM
```



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - ➔ • Sections
  - Tabellenverarbeitung
  - sequentielle Dateien



## warum und weshalb

---

- Vorzug der strukturierten Programmierung ist es, Aufgaben in kleine Module aufteilen zu können
- Umsetzung in COBOL mit Sections
- Aufruf mit `PERFORM section-name`
  - Verzweigung zur Section
  - Ausführen aller Anweisungen in der Section
  - Rücksprung
  - ausführen nächste Anweisung

# Sections

## einfacher Aufruf

---

. . .

**PERFORM VORLAUF**

. . .

**VORLAUF SECTION.**

**[VORLAUF-01.]**

anweisung-1

anweisung-2

continue.

**[VORLAUF-EX.**

**EXIT.]**

## Aufrufmöglichkeiten

---

- PERFORM section I1 TIMES oder
- PERFORM section 20 TIMES
  
- PERFORM section UNTIL bedingung
  
- PERFORM section WITH TEST AFTER UNTIL bedingung

## Syntax Inline-PERFORM

---

### PERFORM

```
[WITH TEST {BEFORE|AFTER}]  
  VARYING {var-1|ind-1}  
  FROM {lit-2|var-2|ind-2}  
  BY {lit-3|var-3}  
  UNTIL bedingung-1
```

```
[anweisung-1]
```

```
[anweisung-2]
```

```
[...]
```

### END-PERFORM

## Syntax Outline-PERFORM

---

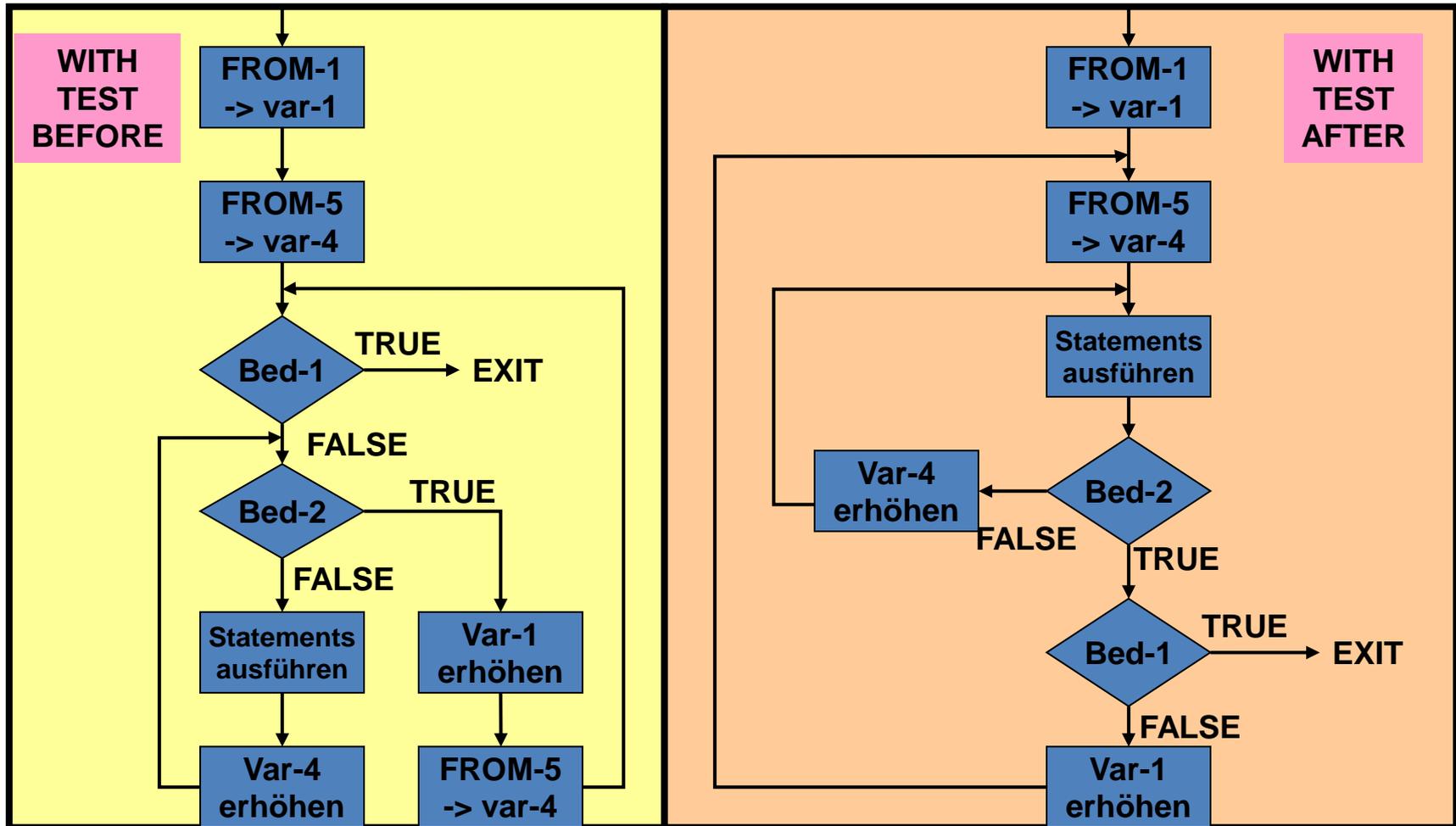
```
PERFORM      section
              [WITH TEST {BEFORE|AFTER}]
              VARYING  {var-1|ind-1}
              FROM     {lit-2|var-2|ind-2}
              BY       {lit-3|var-3}
              UNTIL    bedingung-1
```

## PERFORM mit zwei Laufvariablen

---

```
PERFORM      section
             [WITH TEST {BEFORE|AFTER} ]
             VARYING  {var-1|ind-1}
             FROM     {lit-2|var-2|ind-2}
             BY       {lit-3|var-3}
             UNTIL    bedingung-1
             AFTER    {var-4|ind-4}
             FROM     {lit-5|var-5|ind-5}
             BY       {lit-6|var-6}
             UNTIL    bedingung-2
```

## PERFORM mit zwei Laufvariablen – Ablauflogik



## PERFORM - Bedeutung der Parameter – 1

---

- **WITH TEST BEFORE | WITH TEST AFTER**
  - Diese beiden Abbruchbedingungen haben die gleiche Bedeutung; einmal werden die Tests nach den Statements, einmal vor der Ausführung der Statements ausgeführt.
- **VARYING**
  - Als Parameter gibt man den Subscript oder den Index an, der beim Bearbeiten der Schleife benutzt und verändert werden soll.

## PERFORM - Bedeutung der Parameter – 2

---

- FROM
  - Mit dem Parameter gibt man den Anfangswert an, d.h. den Wert, den der Laufindex vor dem Durchlaufen haben soll.
- BY
  - Nach jedem Durchgang durch die Schleife wird der Laufindex um das hier genannte Inkrement erhöht.

## PERFORM - Bedeutung der Parameter – 3

---

- UNTIL
  - Eine Iteration ist die wiederholte Ausführung eines oder mehrerer Befehle. Die Abbruchbedingung wird durch die UNTIL-Klausel angegeben. Die Schleife wird so lange durchlaufen, bis die Abbruchbedingung erfüllt ist.
- section
  - Beim out-line-PERFORM wird ein Prozedurname angegeben. Die dort kodierten Befehle werden für jeden Iterationsschritt ausgeführt.

## PERFORM - Bedeutung der Parameter – 4

---

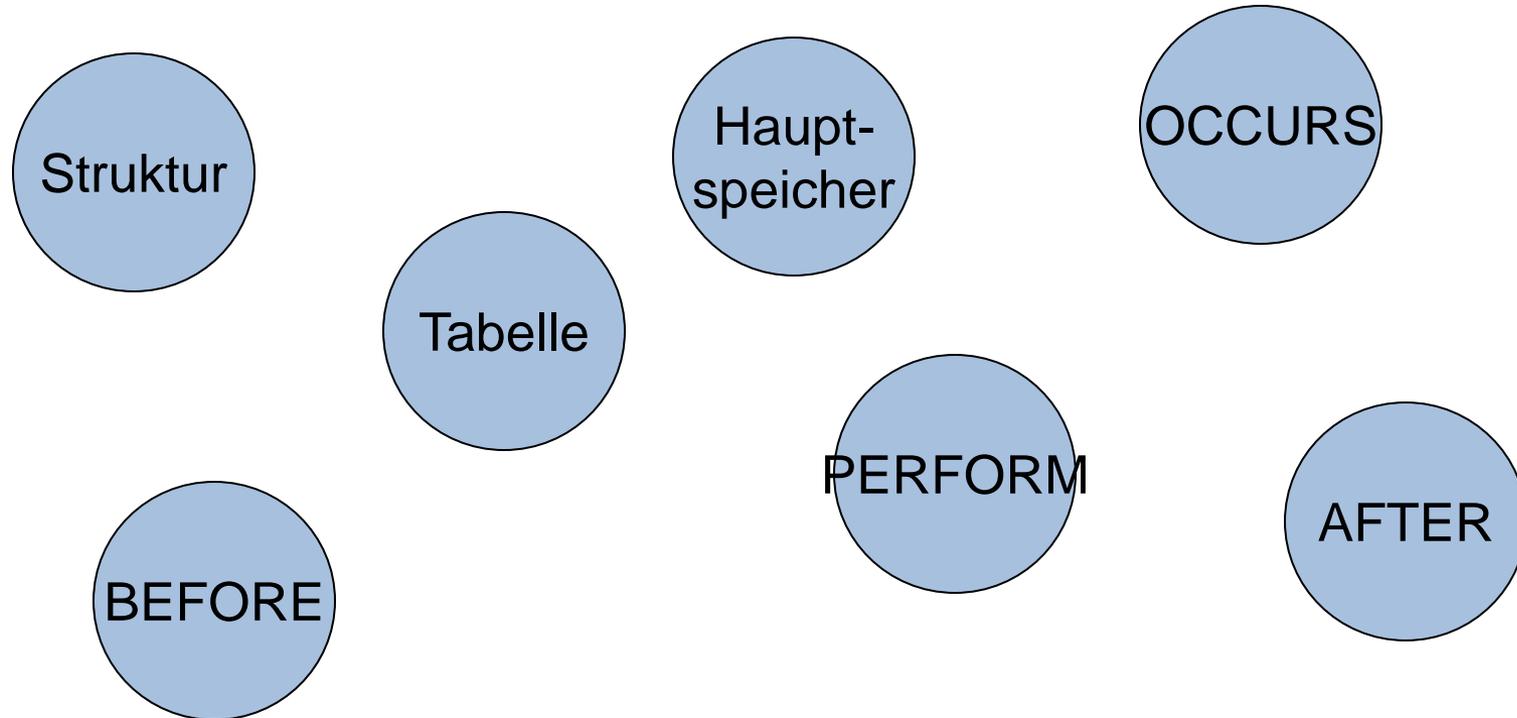
- **PERFORM ... END-PERFORM**
  - Bei dem in-line-PERFORM werden die zwischen dem PERFORM und dem END-PERFORM kodierten Befehle durchlaufen, bis die Abbruchbedingung(en) erfüllt ist (sind).



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - ➔ • Tabellenverarbeitung
  - sequentielle Dateien

## Begriffe

---



## Fragen

---

- Was ist eine Tabelle?
- Wie ist eine Tabelle definiert?
- Welche Möglichkeit der Adressierung gibt es?
- Welche Möglichkeiten der Tabellenverarbeitung gibt es?
- Wie kann man Tabellen mit Anfangswerten versehen?

## Grundlagen

---

- Große Mengen von Daten
- gleichartige Daten
- Unterscheidung nur durch bestimmte Zuordnung
- Gleichheit der Daten in
  - Länge
  - Typ
  - Speicherformat

## Beispiel Tagesumsatz – unbrauchbar

---

01	STAMMSATZ.	
05	UMSATZ-TAG-001	PIC 9(4)V99.
05	UMSATZ-TAG-002	PIC 9(4)V99.
	...	
05	UMSATZ-TAG-366	PIC 9(4)V99.

01 STAMMSATZ.

05 UMSATZ-TAG OCCURS 366  
PIC 9(4)V99.

- Im Hauptspeicher stehen die Daten direkt hintereinander.

## Beispiel Gehaltstabelle

---

01 GEHALTS-TABELLE.

05 GEHALT OCCURS 12.

10 BRUTTO PIC 9(5)V99.

10 ZULAGE PIC 9(5)V99.

10 ABZUG PIC 9(5)V99.

- Im Hauptspeicher stehen die Daten direkt hintereinander in der Form:
  - BRUTTO(1),ZULAGE(1),ABZUG(1),BRUTTO(2) etc.



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien

## Begriffe

---

DIVISION

INPUT-  
OUTPUT

OPEN  
CLOSE

READ  
WRITE

BEFORE

ACCESS

MODE

wo steht was?

---

- **ENVIRONMENT DIVISION**
  - Verbindungsname zur Job Control
  - Programm logischer Name
- **DATA DIVISION**
  - Gestalt der Buffer
  - Satzbeschreibung
- **PROCEDURE DIVISION**
  - OPEN, CLOSE, WRITE, READ, REWRITE, DELETE, START

**FILE-CONTROL.**

**SELECT Dateiname**

**ASSIGN TO DDname-1 [DDname-2] . . .**

**[ORGANIZATION IS SEQUENTIAL]**

**[ACCESS MODE IS SEQUENTIAL]**

**[FILE STATUS IS Datename-3]**

**[PASSWORD IS Datename-4]**

- **ORGANIZATION**
  - logischer Aufbau der Datei
  - Organisation der Datei
    - sequentiell, indiziert -> später, relativ -> später
- **ACCESS MODE**
  - Zugriffsart auf die Sätze der Datei
    - sequentiell, direkt -> später, dynamisch -> später

- **PASSWORD**
  - IBM-Erweiterung -> noch nicht gesehen
- **FILE-STATUS**
  - Definition eines Datenfeldes, in das während der Verarbeitung der Zustand der Ein-/Ausgabe-operation aufgenommen wird
  - 2-stelliges Feld in DATA DIVISION
  - Werte der beiden Bytes siehe COBOL Language Reference “Status Key”

## Befehle – OPEN / CLOSE

---

- OPEN INPUT Dateiname-1 [Dateiname-2]
- OPEN OUTPUT Dateiname-1 [Dateiname-2]
  
- CLOSE Dateiname-1 [Dateiname-2]

## Befehle – READ

---

- READ    dateiname [NEXT] RECORD  
          [INTO Datennamen]  
          [AT END           Anweisung-1]  
          [NOT AT END      Anweisung-2]

END-READ

zusätzlicher Bereich

z.B. Ende-Schalter

z.B. Verarbeitung

## Befehle – WRITE

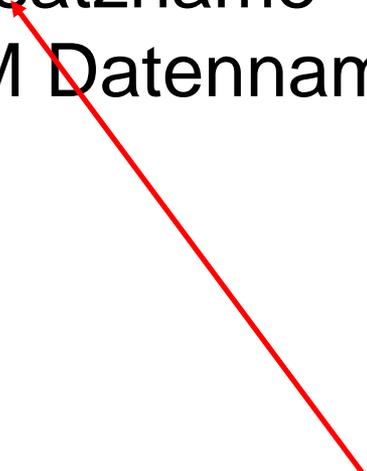
---

- WRITE datensatzname  
[FROM Datenname]  
END-WRITE

zusätzlicher Bereich



Stufennummer 01  
in FILE SECTION



## Programmlogik – Teil 1

---

```
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SPECIAL-NAMES.  
    DECIMAL-POINT IS COMMA.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT  AMT-EINGABE ASSIGN TO DDSE01.  
    SELECT  AMT-AUSGABE ASSIGN TO DDSA01.  
  
*  
DATA DIVISION.  
FILE SECTION.  
* DATEI DER AEMTER  
FD  AMT-EINGABE  
    BLOCK CONTAINS 0 RECORDS  
    RECORDING MODE IS F.  
01  AMT-EIN-SATZ                                PIC  X(80) .  
***  
FD  AMT-AUSGABE  
    BLOCK CONTAINS 0 RECORDS  
    RECORDING MODE IS F.  
01  AMT-AUS-SATZ                                PIC  X(80) .
```

## Programmlogik – Teil 2

---

WORKING-STORAGE SECTION.

01 EINGABE-SATZ .

02 EINGABE-ZEILE PIC X(200) OCCURS 200.

/\*\*\*\*\*

PROCEDURE DIVISION.

SCHEIN SECTION.

\*\*\*\*\*

OPEN INPUT AMT-EINGABE

OPEN OUTPUT AMT-AUSGABE

READ AMT-EINGABE INTO EINGABE-ZEILE(1)

MOVE EINGABE-ZEILE(1) TO AMT-AUS-SATZ

WRITE AMT-AUS-SATZ

CLOSE AMT-EINGABE

CLOSE AMT-AUSGABE

## Programmlogik – Zusammenfassung

---

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT **AMT-EINGABE** ASSIGN TO DD01.

DATA DIVISION.

FILE SECTION.

FD **AMT-EINGABE**,

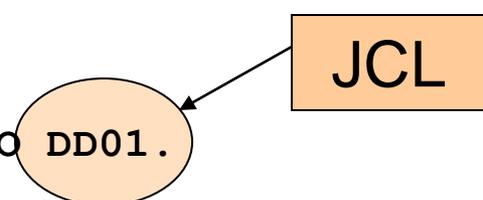
01 **AMT-EIN-SATZ** PIC X(80).

PROCEDURE DIVISION.

OPEN INPUT **AMT-EINGABE**

READ **AMT-EINGABE**

aber: WRITE **AMT-AUS-SATZ** !!



JCL

# zu guter Letzt

das gibt es auch

---

- GO TO NEIN!!!!
- NEXT SENTENCE NEIN!!!!
- Dringende Empfehlung:  
>>> Punkt nur am Ende einer Section <<<



- 
- Einführung
  - Programmaufbau oder „Das erste Programm!“
  - Arbeiten mit einfachen numerischen Daten
  - Schleifen mit Zähler
  - Entscheidungen
  - Schleifen mit Bedingungen
  - Sections
  - Tabellenverarbeitung
  - sequentielle Dateien

