Programmieren 1

ptlwedel

PRIVATE BERUFSFACHSCHULE

Wintersemester 2016/2017

Marcus Riemer, B.Sc.

Basierend auf den Unterlagen „Programmstrukturen 1“
von Prof. Dr. Andreas Häuslein

0.1 Marcus Riemer

Marcus Riemer, B.Sc.
Raum M14, Audimax

Web : FH-Wedel.de/~mri
Mail : mri@ptl.de

0.1 Marcus Riemer

Marcus Riemer, B.Sc.
Raum M14, Audimax

Jetzt eingesprungen:
“Programmieren 1” mit Dieter Opitz, Christian Krug, Birger Wolter

Später auch in:

- Programmieren 2? (2. Sem)
- Web-Anwendungen (4. Sem)
- Übg. Objektorientierte Programmstrukturen (4. Sem)
- Prakt. Computergrafik (4. Sem)

0.1 Weiterer Verlauf dieses Semester

- Vorlesung und Übung zu den im Stundenplan angegebenen Zeiten, allerdings von jetzt an grundsätzlich in RZ3.
- Ablauf wie in Office-Software: Zuhören und praktische Arbeit im Wechsel.
- Fragen grundsätzlich jederzeit.

1.1 Grundkonzepte der Datenverarbeitung

Notwendige Angaben zur Realisierung der Datenverarbeitung:

Eingabe

- Angaben zu den einzugebenden Daten
  - Um welche Art von Daten (Wertebereiche) handelt es sich?
  - Wie kommen sie in das Programm?

Verarbeitung

Programm

- Angaben zu auszuführenden Verarbeitungsschritten (Programmbefehle, Anweisungen)
  - Welcher Art sind die notwendigen Verarbeitungsschritte?
  - In welcher Abfolge sollen die Verarbeitungsschritte durchlaufen werden?

Ausgabe

- Angaben zu auszugebenden Daten
  - Um welche Art von Daten handelt es sich?
  - Wo werden die Daten gespeichert?

1.2 Grundkonzepte der Programmierung

Unser Vorgehen im Überblick:

Problem-/Aufgabenstellung verstehen

↓

Verarbeitungsverfahren entwickeln
(Algorithmus)

↓

Verfahren in ein Computer-Programm umsetzen
(Programmieren)

↓

Computer-Programm laufen lassen,
Ergebnisse wahrnehmen und nutzen

1.3 Algorithmische Grundstrukturen

Sequenz

- Einfache Abfolge von Aktionen, beliebig lang
- Darstellung als Struktogramm:

Aktion 1
Aktion 2
Aktion 3

Beispiel:

Multipliziere Zähler1 mit Zähler2
Multipliziere Nenner1 mit Nenner2

- Darstellung in Pseudo-Code:

```
Anfang
    Aktion 1
    Aktion 2
    Aktion 3
Ende
```

```
begin
    Aktion 1
    Aktion 2
    Aktion 3
end
```

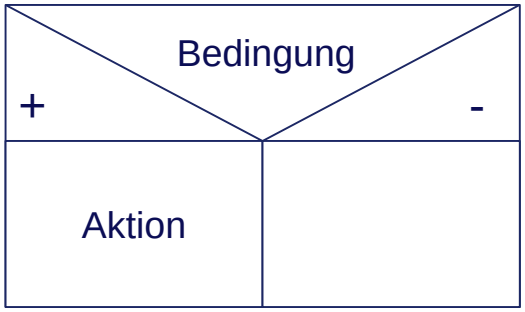
Beispiel:

```
begin
    Multipliziere Zähler1 mit Zähler2
    Multipliziere Nenner1 mit Nenner2
end
```

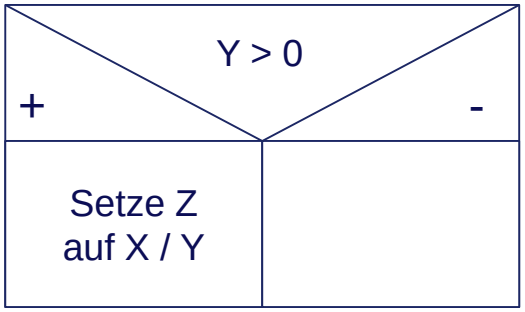
1.3 Algorithmische Grundstrukturen

Bedingte Aktion

- Aktion wird ausgeführt, wenn eine Bedingung gilt
- Wenn die Bedingung nicht gilt, wird *keine* Aktion ausgeführt
- Darstellung als Struktogramm:



Beispiel:



- Darstellung in Pseudo-Code:

Wenn Bedingung
dann Aktion

if Bedingung
then Aktion

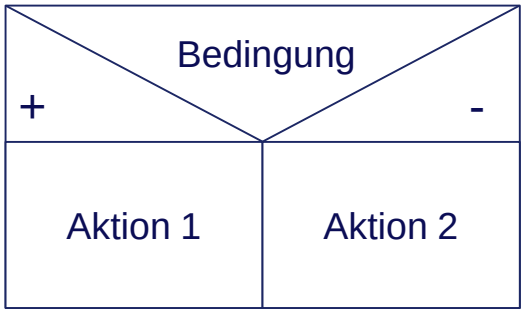
Beispiel:

if Y > 0
then Setze Z auf X / Y

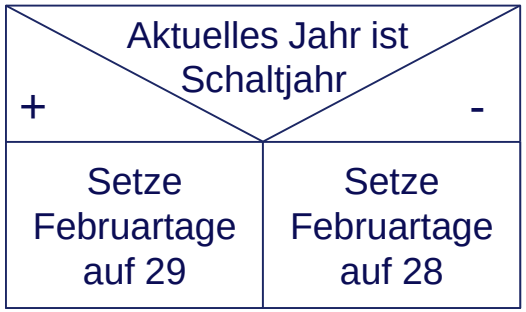
1.3 Algorithmische Grundstrukturen

Alternative

- Eine von zwei möglichen Aktionen wird ausgeführt, welche das ist, hängt von einer Bedingung ab
- Darstellung als Struktogramm:



Beispiel:



- Darstellung in Pseudo-Code:

Wenn Bedingung
dann Aktion 1
sonst Aktion 2

if Bedingung
then Aktion 1
else Aktion 2

Beispiel:

if Aktuelles Jahr ist Schaltjahr
then Setze Februartage auf 29
else Setze Februartage auf 28

Mehrfachauswahl (Fallunterscheidung)

- Auswahl unter mehreren möglichen Aktionen anhand einer Auswahlgröße
- Jede Aktion ist mit möglichen Werten der Auswahlgröße markiert
- Eine Aktion wird dann ausgeführt, wenn ihre Markierung mit dem aktuellen Wert der Auswahlgröße übereinstimmt
- Darstellung als Struktogramm:

Auswahlgröße				
Mar-kierung 1	Mark. 2		Mark. n	Sonst
Aktion 1	Aktion 2	...	Aktion n	Aktion Sonst

Beispiel:

Monat				
1, 2,12	3 - 5	6 - 8	9 - 11	Sonst
Jahreszeit auf "Winter" setzen	Jahreszeit auf "Frühling" setzen	Jahreszeit auf "Sommer" setzen	Jahreszeit auf "Herbst" setzen	Fehler-meldung

- Darstellung in Pseudo-Code:

```
case Auswahlgröße
  Markierung 1: Aktion 1
  Markierung 2: Aktion 2
  Markierung 3: Aktion 3
  else Aktion 4
endcase
```

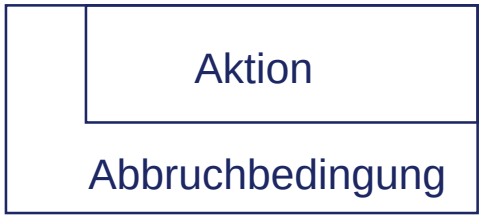
Beispiel:

```
case Monat
  1, 2, 12: Jahreszeit auf "Winter" setzen
  3 - 5:    Jahreszeit auf "Frühling" setzen
  6 - 8:    Jahreszeit auf "Sommer" setzen
  9 - 11:   Jahreszeit auf "Herbst" setzen
  else Fehlermeldung
endcase
```

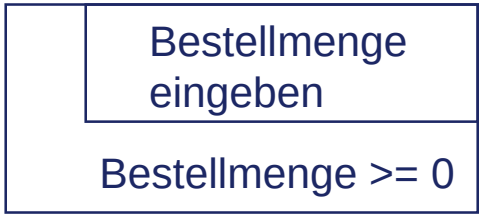
1.3 Algorithmische Grundstrukturen

Iteration (Wiederholung) mit Abbruchbedingung

- Ausführen einer Aktion
- Wiederholen der Aktion, sofern Abbruchbedingung *nicht* gilt
- Ende der Wiederholung, sobald Abbruchbedingung gilt
- Darstellung als Struktogramm:



Beispiel:



- Darstellung in Pseudo-Code:

Wiederhole Aktion
Bis Abbruchbedingung

do Aktion
until Abbruchbedingung

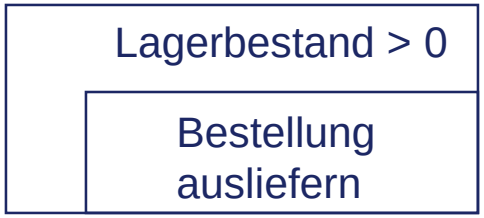
Beispiel: **Wiederhole** Bestellmenge eingeben
Bis Bestellmenge >= 0

1.3 Algorithmische Grundstrukturen

Iteration (Wiederholung) mit Eintrittsbedingung

- Nur wenn Eintrittsbedingung gilt, Aktion ausführen
- Aktion wiederholen, solange Eintrittsbedingung gilt
- Ende der Wiederholung, sobald nach der Aktion Eintrittsbedingung nicht mehr gilt
- Darstellung als Struktogramm:

Beispiel:



- Darstellung in Pseudo-Code:

Solange Bedingung
Wiederholen Aktion

while Bedingung
do Aktion
endwhile

Beispiele:

Solange Lagerbestand > 0
Wiederholen Bestellung ausliefern

while Lagerbestand > 0
do Bestellung ausliefern
endwhile

Das sind alle!

Bild: Futurama™, ©FOX, Grabpic von futurama-france.fr

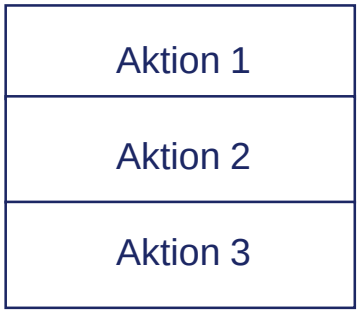
1.3 Algorithmische Grundstrukturen

Fortgeschrittenen Konzepte sind Varianten dieser Strukturen

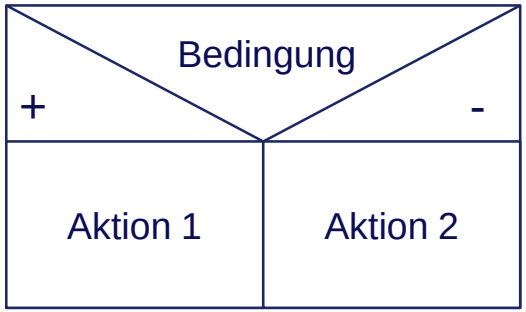
- Objektorientierung Progammierung
- Funktionale Programmierung
- Ereignisorientierte Programmierung
- …

Jeder Algorithmus kann durch die Kombination der Strukturen `Aktion`, `Auswahl` und `Wiederholung` ausgedrückt werden.

`Aktion`



`Auswahl`

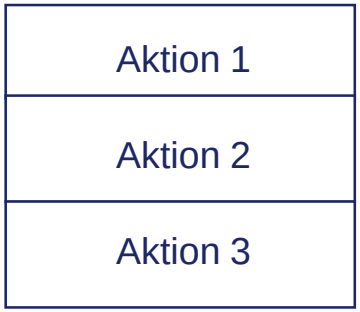


`Wiederholung`

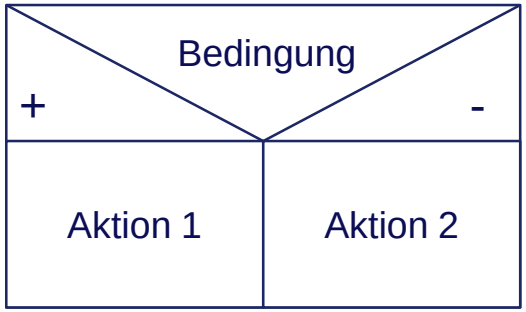
1.3 Algorithmische Grundstrukturen

Jeder Algorithmus kann durch die Kombination der Strukturen `Aktion`, `Auswahl` und `Wiederholung` ausgedrückt werden.

`Aktion`



`Auswahl`



`Wiederholung`

Wie werden die folgenden Strukturen dargestellt?

- Bedingte Aktion “Absturz” unter der Bedingung “Windows?”
- Mehrfachauswahl für Auswahlgröße “Rasse” mit Ausprägungen “Zerg”, “Terraner”, “Protoss”?
- Mindestens einmalige Ausführung der Aktion “Lernen”, Abbruch sobald die Bedingung “Alles gelernt” erfüllt ist.

1.3 Algorithmische Grundstrukturen

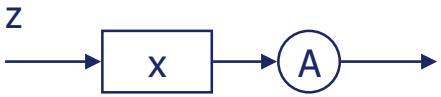
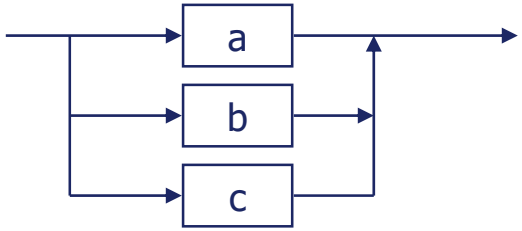
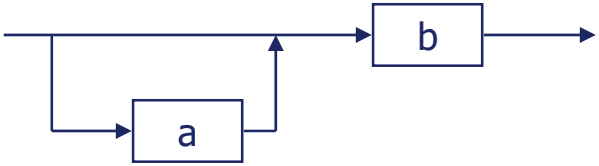
Warum braucht es dann fortgeschrittene Konzepte?

- Kompaktere Darstellung von speziellen Sachverhalten
- Bessere Wiederverwendbarkeit durch Kapselung

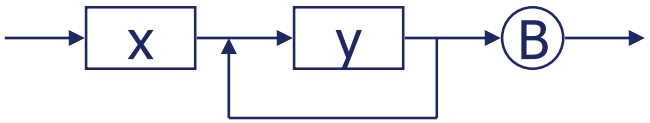
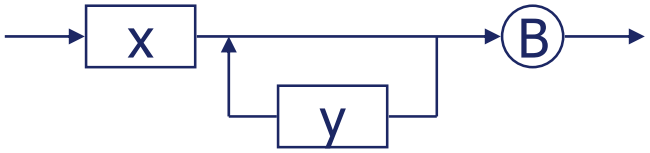
Warum sollte ich jetzt noch zuhören?

- Kombination von einfachen Dingen kann komplexe Konstrukte ergeben
- Programme werden häufiger gelesen als geschrieben, Kommunikation ist also entscheidend.
- Programme in der “echten Welt” verwenden diese Konzepte.

1.4 Syntaxdiagramme und Backus-Naur-Form im Vergleich

Bezeichnung	BNF	Syntaxdiagramm
Terminales Symbol	A	
Nichtterminales Symbol	$\langle A \rangle$	
Sequenz	$\langle a \rangle AB \langle b \rangle$	
Regel	$\langle z \rangle ::= \langle x \rangle A$	
Alternativen	$\langle a \rangle \mid \langle b \rangle \mid \langle c \rangle$	
Option	$[\langle a \rangle] \langle b \rangle$	

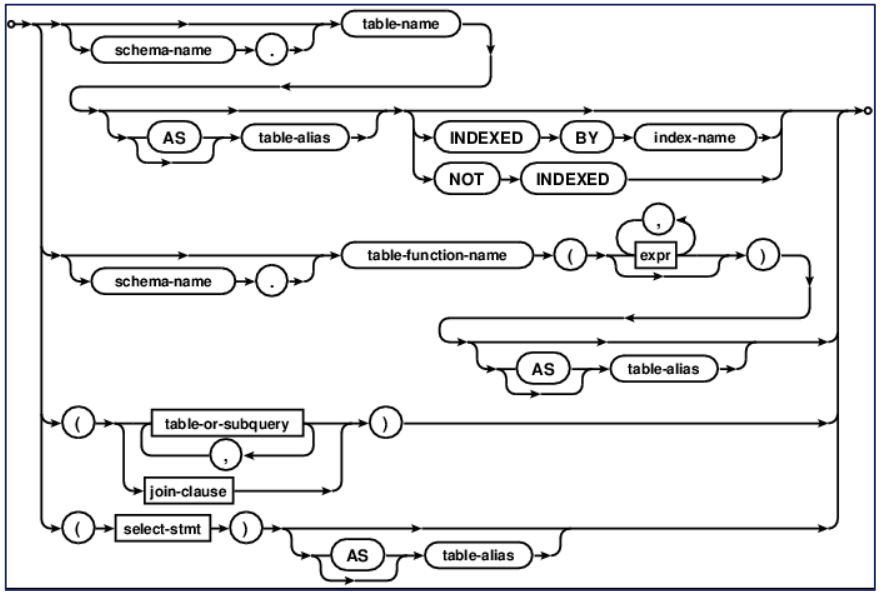
1.4 Syntaxdiagramme und Backus-Naur-Form im Vergleich

Bezeichnung	BNF	Syntaxdiagramm
Wiederholung	$\langle x \rangle \{ \langle y \rangle \}_1^* B$	
Wiederholung mit Option	$\langle x \rangle \{ \langle y \rangle \}_0^* B$	

1.4 Anwendung Syntaxdiagramme & BNF

Wozu brauche ich Syntaxdiagramme?

- Validierung von Eingaben muss nicht als „normaler“ Code geschrieben werden, stattdessen ist die BNF wesentlich besser lesbar.
- Reale Programmiersprachen und Dateiformate werden in diesen Formaten beschrieben.



Beschreibung der Syntax für Tabellen oder Unterabfragen in SQLite
https://www.sqlite.org/lang_select.html

1.5 Übungsaufgabe „Telefonnummer“

Beschreiben sie die Struktur von Telefonnummern anhand der folgenden Merkmale in der Backus-Naur-Form oder als Syntax-diagramm. Das Nichtterminalsymbol **`Ziffer`** können sie als gegeben voraussetzen.

- Eine **`Telefonnummer`** beginnt optional mit einer **`Vorwahl`**, auf die eine **`Teilnehmernummer`** folgt.
- Die **`Vorwahl`** besteht aus zwei Teilen:
  - Die Ländervorwahl “00” oder “+” und einem zweistelligen Ländercode aus **`Ziffern`**.
  - Sofern keine Ländervorwahl existiert beginnt die drei- oder vier-stellige Ortsvorwahl mit “0”, anderenfalls entfällt die “0”.
- Die **`Teilnehmernummer`** besteht aus mindestens einer **`Ziffer`**.
  - Optional wird eine **`Teilnehmernummer`** von einer **`Durchwahl`** aus beliebig vielen **`Ziffern`** gefolgt, welche durch das “-” Zeichen eingeleitet wird.

1.6 Übungsaufgabe „Klammersprache“

Finden sie heraus welche der folgenden Texte auf die beschriebene Backus-Naur-Form passen. Wandeln sie diese Darstellung dafür zunächst in das äquivalente Syntaxdiagramm um. Die Auswertung beginnt dabei mit der Regel „*Ausdruck*“.

$$Ausdruck \ ::= \langle Zahl \rangle \mid "(" \ \langle Ausdruck \rangle \ \langle Operator \rangle \ \langle Ausdruck \rangle \ ")"$$
$$Zahl \ ::= \{\langle Ziffer \rangle\}_1^*$$
$$Ziffer \ ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"$$
$$Operator \ ::= " + "|" - "|" * "|"mod"|"div"$$

a) 4
b) 4 + 2
c) (32 + 17)
d) (4 * 3)
e) (2 * 2 + 3)
f) -42
g) (32 -(21 * 2))
h) (2)
i) (3 mod 7)
j) (42 / 31)
k) (2^3)
l) (1+(2+(3+4)))
m) (((2*2)+1)
n) ((1+1)+(2+2))
o) ((1+2)+3))