Die Endnutzerprogrammiersprache o++o
auf 25 Seiten

Klaus Benecke

Inhaltsverzeichnis

1 Einleitung 1
2 Was ist eine strukturierte Tabelle? 2
3 Beispiele 6
3.1 Einzeilige Berechnungen 6
3.2 Diagramme 8
3.3 Schulprobleme 12
3.4 Ein 4 zeiliges o++o-Programm, für das EXCEL mehr als sechs Arbeitsblätter benötigt! 13
3.5 Tabellenkalkulation mit o++o 15
3.6 Strukturierte Dokumente 17
3.7 Ein Anfangswertproblem einer Differentialgleichung 20
4 Die Designprinzipien von o++o 21
5 Danksagung 25

1 Einleitung

o++o ist eine leistungsfähige und gleichzeitig einfach anzuwendende Computersprache (Sprache für OttoNormalVerbraucher). Aufgrund leistungsfähiger Funktionen zur Berechnung, Textbearbeitung und Strukturierung, die auch auf Datenmengen in einem weiten Bereich - hinsichtlich Umfang und Komplexität - angewendet werden können, ergeben sich vielfältige Einsatzgebiete.

Insbesondere die Analyse von Daten kann durch den Einsatz von o++o flexibler, zielgerichteter, genauer und produktiver werden. So können Daten aus verschiedenen Quellen zusammengefasst, selektiert und in die gewünschte Struktur überführt werden. Die Ergebnisse sind ohne weitere Zwischenschritte grafisch darstellbar. Für die Visualisierung sind vielfältige Darstellungsformen wählbar.

Da o++o-Programme sehr kompakt sind, können durch Eingabe weniger Programmzeilen gezielt verschiedene Szenarien untersucht werden, die Wirkung der Änderung von Parametern bestimmt werden oder Prognosen erstellt werden.

Der Berechnungsablauf und die verwendeten Algorithmen werden durch das o++o-Programm vollkommen transparent und nachvollziehbar abgebildet. Somit wird auch die Vertrauenswürdigkeit der Ergebnisse signifikant erhöht.

o++o garantiert dem Anwender vollständige Kontrolle über seine Berechnungen, Analysen, Dokumente und Visualisierungen.

Aus diesen Merkmalen ergibt sich ein vielfältiges Anwendungsspektrum. Beispiele sind:

- Aufbereitung und Analyse von Geschäftsdaten
- Klassifizierung und Auswertung von Messdaten (z.B. Umwelt, Verkehr, Energie, Medizin)

- Diagnose technischer Systeme, Leistungs- und Qualitätsoptimierung
- Prognosen

Weiterhin bietet o++o die Möglichkeit, Dokumente zu analysieren.

Aufgrund seiner Einfachheit und Vielseitigkeit stellt o++o auch für Privatanwender ein nützliches Produkt dar. Dieses unterscheidet sich von der Vielzahl kommerzieller Produkte für Privatanwender auf dem Markt darin, dass es nicht auf einen bestimmten Anwendungsfall reduziert ist.

Als erstes muss der Benutzer verstehen, was ein Schema ist und was die Tabellen oder Dokumente sind, die zu diesem Schema gehören. Dann wird es nicht zu schwer sein, die Programme für Selektion, Berechnung und Umstrukturierung der ersten Beispiele zu erfassen. Alle Operationen erlauben eine kompakte und gut lesbare Formulierung von (komplexen) Anfragen. Sie gelten für verschachtelte Listen oder Mengen und sie sind neu in der Datenbankwelt. Berechnungen werden stets auf alle entsprechenden Elemente angewandt. Daher können auch an dieser Stelle Programmschleifen vermieden werden. Die Umstrukturierung mit der gib-Klausel ist sehr ausdrucksstark, da sie mit Sortieren (m, b), Duplikate-Eliminierung (m), Aggregation (++, min, max, ++1, ++:, ||, &&, **,variance) kombiniert wird. Wir kennen keine andere Umstrukturierungsoperation in einem kommerziellen Produkt, die es erlaubt, eine gegebene Hierarchie nur durch Angabe eines Schemas oder des TTs (TabmentTyp) der gewünschten Struktur umzuwandeln. Obwohl die o++o-Operationen in den Beispielen nur sequentiell angewendet werden, decken sie ein weites Anwendungsgebiet ab.

2 Was ist eine strukturierte Tabelle?

Eine (flache) Tabelle ist eine in Zeilen und Spalten gegliederte rechteckige Anordnung von Zahlen bzw. Daten. In der Regel besitzt jede Spalte einen Namen. Bei einer strukturierten Tabelle besteht der Kopf nicht nur aus einfachen Spaltennamen, sondern aus einem Schema, welches die Strukturierung der Daten beschreibt. Neben dem Komma sind beispielsweise auch Kollektionssymbole für Mengen (m), und Listen (l) zugelassen.

```
ABTEILUNG,  CHEF, (NAME,  GEHALT m), (PROJEKT, BUDGET m)l
Produktion  Ines   Georg  5'000       DB1.0    70'000
                   Paul   6'000       PS1.0    20'000
Entwicklung Johann Clara  5'500       DB2.0    60'000
                   Eric   4'500       KI1.0    50'000
                   Rainer 4'900
Vertrieb    Karl   Ingrid 6'100       DB0.5    40'000
                   Sabine 4'200       PS0.7    30'000
```

abteilungen.tab

abteilungen.tab ist eine strukturierte Tabelle von 3 Abteilungen im tab-Format, wobei die Entwicklungsabteilung 3 Mitarbeiter und 2 Projekte enthält. Da die Menge der (NAME, GEHALT)-Paare unabhängig von der Menge der (PROJEKT,BUDGET)-Paare ist, besteht in dieser Tabelle kein Zusammenhang zwischen den Angestellten und den Projekten. D.h. z.B., dass *Clara* nicht unbedingt am Projekt Datenbanken (DB2.0) arbeiten muss, obwohl die Daten in einer Zeile stehen. *Clara* steht in Beziehung zu *Entwicklung* und *Entwicklung* steht in Beziehung zu *DB2.0* und trotzdem steht *Clara* nicht in Beziehung zu *DB2.0*. Diese Situation wird klarer, wenn man sich vorstellt, dass es sich bei den Projekten um externe Projekte jeder Abteilung handelt. Die Trennung zwischen Mitarbeitern und Projekten wird in der web-Darstellung des Tabments besser verdeutlicht:

ABTEILUNG	CHEF	(NAME,GEHALT m)		(PROJEKT,BUDGET m)	
Produktion	Ines	NAME	GEHALT	PROJEKT	BUDGET
		Georg	5000	DB1.0	70000
		Paul	6000	PS1.0	20000
Entwicklung	Johann	NAME	GEHALT	PROJEKT	BUDGET
		Clara	5500	DB2.0	60000
		Eric	4500	KI1.0	50000
		Rainer	4900		
Vertrieb	Karl	NAME	GEHALT	PROJEKT	BUDGET
		Ingrid	6100	DB0.5	40000
		Sabine	4200	PS0.7	30000

Im Rahmen des relationalen Datenmodells müsste diese Informationsmenge durch 3 flache Tabellen dargestellt werden:

ABTEILUNG,	CHEF m
Produktion	Ines
Entwicklung	Johann
Vertrieb	Karl

(abteilungen1.tab)

ABTEILUNG,	NAME,	GEHALT m
Produktion	Georg	5'000
Produktion	Paul	6'000
Entwicklung	Clara	5'500
Entwicklung	Eric	4'500
Entwicklung	Rainer	4'900
Vertrieb	Ingrid	6'100
Vertrieb	Sabine	4'200

(mitarbeiter1.tab)

ABTEILUNG,	PROJEKT,	BUDGET m
Produktion	DB1.0	70'000
Produktion	PS1.0	20'000
Entwicklung	KI1.0	50'000
Entwicklung	DB2.0	60'000
Vertrieb	DB0.5	40'000
Vertrieb	PS0.7	30'000

(projekte1.tab)

Die Informationen von abteilungen1.tab und mitarbeiter1.tab können mit o++o zu folgender flacher Tabelle zusammengeführt werden:

```
ABTEILUNG,  CHEF,   NAME,  GEHALT m
Produktion  Ines    Georg  5'000
Produktion  Ines    Paul   6'000
Entwicklung Johann  Clara  5'500
Entwicklung Johann  Eric   4'500
Entwicklung Johann  Rainer 4'900
Vertrieb    Karl    Ingrid 6'100
Vertrieb    Karl    Sabine 4'200
```

Ein Ausblenden der Wiederholungen von (ABTEILUNG, CHEF) ist im Rahmen des Relationalen Datenmodells jedoch nicht möglich. Auch kann die Ausgangstabelle mit SQL nicht wieder aus den 3 flachen Tabellen erzeugt werden. In EXCEL kann man zwar strukturierte Tabellen wie abteilungen.tab darstellen. Man kann die Daten aber nicht unmittelbar verarbeiten. Selbst ein Sortieren der Daten ist problematisch, da immer nur flache Tabellen sortiert werden können.

Einfach durch Positionieren der gewünschten Spaltennamen an die ersten Stellen einer Kollektion kann mit o++o sortiert werden:

Programm 1: Sortiere in 3 Kollektionen gleichzeitig

abteilungen.tab

```
gib ABTEILUNG,(GEHALT,NAME m),(BUDGET,PROJEKT m-) m
```

Ergebnis:

```
ABTEILUNG, (GEHALT, NAME  m),(BUDGET, PROJEKT  m-) m
Entwicklung 4500    Eric      60000   DB2.0
            4900    Rainer    50000   KI1.0
            5500    Clara
Produktion  5000    Georg     70000   DB1.0
            6000    Paul      20000   PS1.0
Vertrieb    4200    Sabine    40000   DB0.5
            6100    Ingrid    30000   PS0.7
```

Betrachten wir eine weitere strukturierte Tabelle:

```
FACH,        NOTEl m
Mathematik   2 1 3
Physik       2 2 3
Englisch     1 4
Deutsch      2 1 3 3
Chemie       1 1 2 4 1
```

(meinenoten.tabh)

Hier ist zu jedem Fach eine Liste von Noten gegeben. Die Listenelemente - die Noten - wurden jedoch horizontal angeordnet (tabh-Format). Wenn wir die gleiche Informationsmenge durch eine flache Tabelle repräsentieren würden, ergäbe sich die folgende unübersichtliche Darstellung:

Programm 2: Stelle meinenoten.tabh in einer flachen Tabelle dar!

meinenoten.tabh

```
gib FACH,NOTE l
```

Ergebnis (tab):

```
FACH,       NOTE  l
Chemie      1
Chemie      1
Chemie      2
Chemie      4
Chemie      1
Deutsch     2
Deutsch     1
Deutsch     3
Deutsch     3
Englisch    1
Englisch    4
Mathematik  2
Mathematik  1
Mathematik  3
Physik      2
Physik      2
Physik      3
```

Es sei angemerkt, dass diese Liste von (FACH, NOTE)-Paaren keine Relation im Sinne des Relationalen Datenmodells ist. Eine Relation ist eine Menge. Durch Ersetzen von l durch m im gib-Teil würde eine solche entstehen. Damit entfielen jedoch alle doppelten Zeilen.

Stellt man einfach die Ausgangstabelle mit dem Schema (FACH,NOTEl m) im tab-Format dar, so stimmt die Darstellung mit der obigen überein. Lediglich erscheint jedes Fach nur einmal:

```
FACH,       NOTEl m
Chemie      1
            1
            2
            4
            1
Deutsch     2
            1
            3
            3
Englisch    1
            4
Mathematik  2
            1
            3
Physik      2
            2
            3
```

(meinennoten.tab)

D.h., die gleiche Informationsmenge kann einmal in 6 Zeilen und einmal in 18 Zeilen dargestellt werden. Damit wird klar, dass man für die Repräsentation von Tabellen mehrere Formate benötigt. Mit o++o kann man diese Tabelle sogar im Dokumentformat xml - einfach durch einen Mausklick - repräsentieren:

```
<MEINENOTEN>
  <FACH>Chemie</FACH>
  <NOTE>1</NOTE>
  <NOTE>1</NOTE>
  <NOTE>2</NOTE>
```

```
   <NOTE>4</NOTE>
   <NOTE>1</NOTE>
   <FACH>Deutsch</FACH>
   <NOTE>2</NOTE>
   <NOTE>1</NOTE>
   <NOTE>3</NOTE>
   <NOTE>3</NOTE>
   <FACH>Englisch</FACH>
   <NOTE>1</NOTE>
   <NOTE>4</NOTE>
   <FACH>Mathematik</FACH>
   <NOTE>2</NOTE>
   <NOTE>1</NOTE>
   <NOTE>3</NOTE>
   <FACH>Physik</FACH>
   <NOTE>2</NOTE>
   <NOTE>2</NOTE>
   <NOTE>3</NOTE>
</MEINENOTEN>
```

In diesem Fall enthält die Repräsentation zwar sehr viel Redundanz bei den Metadaten (Spaltennamen) und ist daher noch schlechter lesbar. Bei vielen Dokumenten ist diese Repräsentation von strukturierten Dokumenten jedoch vorteilhaft gegenüber einer tabellarischen Darstellung. Wenn es viele Darstellungen einer Datenmenge gibt, sprechen wir auch von einer Abstraktion. Wir nennen allgemeine strukturierte Tabellen und strukturierte Dokumente auch Tabmente. Tabmente sind also eine Abstraktion von Tabellen und Dokumenten, die in Form von tab, tabh, xml, csv, json, hsq, hsqh, ment und html - Dateien dargestellt werden können.

3 Beispiele

Wir stellen zunächst einige numerische Berechnungen vor, dann folgen Beispiele zu strukturierten Tabellen und zuletzt Beispiele, die strukturierte Dokumente anfragen. Man kann o++o von ottops.de/news downloaden und die folgenden Programme testen. Da von oben nach unten und von links nach rechts gerechnet wird, kann man durch Verschieben des Kommentarzeichens # die schrittweisen Berechnungen gut nachvollziehen.

3.1 Einzeilige Berechnungen

Wir unterscheiden binäre (zweistellige) Operationen, wie die Addition +. Diese werden stets zwischen die beiden Inputwerte geschrieben. In *3 + 4* ist *3* der erste Inputwert und *4* der zweite. Unäre (einstellige) Operationen werden in o++o stets nach dem Inputwert geschrieben.

4 sqrt

ergibt beispielsweise 2. .

16 sqrt sqrt

ist die Wurzel aus der Wurzel von 16 also auch 2. .

Ternäre (dreistellige) Operationen werden ebenfalls nach dem ersten Inputwert geschrieben. Da das Komma bereits als Paarbildungsoperation vergeben ist, wird in o++o das Ausrufezeichen an vielen Stellen als Trennzeichen benutzt. Bei ternären Operationen trennt es den zweiten vom dritten Inputwert. .. ist eine zweistellige aber ... ist eine dreistellige Operation.

0 ...20!2 ergibt beispielsweise die Liste

0 2 4 6 8 10 12 14 16 18 20

Programm 3: Division mit verbesserter Lesbarkeit	Ergebnis
`1:7 '3 # '3: unär`	`0.142'857'142'857`

Programm 4: Division mit Rundung	Ergebnis
`1:7 rnd 3 # rnd: binär`	`0.143`

Programm 5: Potenzieren	Ergebnis
`3 hoch 20 '3 # hoch: binär`	`3'486'784'401`

Programm 6: Addition gebrochener Zahlen	Ergebnis
`3/4 + 1/3`	`13/12`

Programm 7: Typ des ersten Eingabewerts bleibt erhalten	Ergebnis
`3/4 + 0.3`	`21/20`

Programm 8: Typ des ersten Eingabewerts bleibt erhalten	Ergebnis
`0.3 + 3/4`	`1.05`

Programm 9: Differenz	Ergebnis
`3 - 2`	`1`

Programm 10: Sinus von pi : 2	Ergebnis
`pi : 2 sin # pi: Konstante; sin: unär`	`1.`

Programm 11: Sinus von 30 Grad	Ergebnis
`30:180*pi sin`	`0.5`

Programm 12: Wie viele 10-stellige Binärzahlen gibt es?	Ergebnis
`2 hoch 10`	`1024`

Programm 13: Berechne die Kantenlänge eines Würfels mit dem Volumen 2!	Ergebnis
`2 hoch 1/3`	`1.25992104989`

Programm 14: Summe von 4 Zahlen	Ergebnis
`3.21 4.56 6.88 9.32 ++ # ++ :unär`	`23.97`

++ ist unär, da alle Zahlen als **eine** Liste aufgefasst werden.

Programm 15: Summe der Zahlen von 1 bis 100	Ergebnis
`1 .. 100 ++          # .. : binär: von .. bis`	`5050`

Programm 16: Produkt der Zahlen von 10 bis 40	Ergebnis
`10 .. 40 **   # ** : unär`	`22484437920191185360053220612767744000000000`

Man erkennt am Ergebnis, dass man mit o++o beliebig große ganze Zahlen verarbeiten kann.

Programm 17: Maximum von Zahlen	Ergebnis
`1/3 2/7 max     # 1/3 2/7 ist eine Liste von 2 Zahlen`	`1/3`

Programm 18: Durchschnitt von mehreren Noten	Ergebnis
`1 3 2 1 3 4 ++:`	`2.33333333333`

Programm 19: Werte 2 Terme in 2 getrennten Spalten aus!
`FLAECHE:= 2.34*1.23     # := : Zuweisung; FLAECHE wird Spaltenname` `UMFANG := 2.34+1.23*2   # Erweiterung um eine Spalte UMFANG`
Ergebnis
`FLAECHE, UMFANG`
`2.8782   7.14`

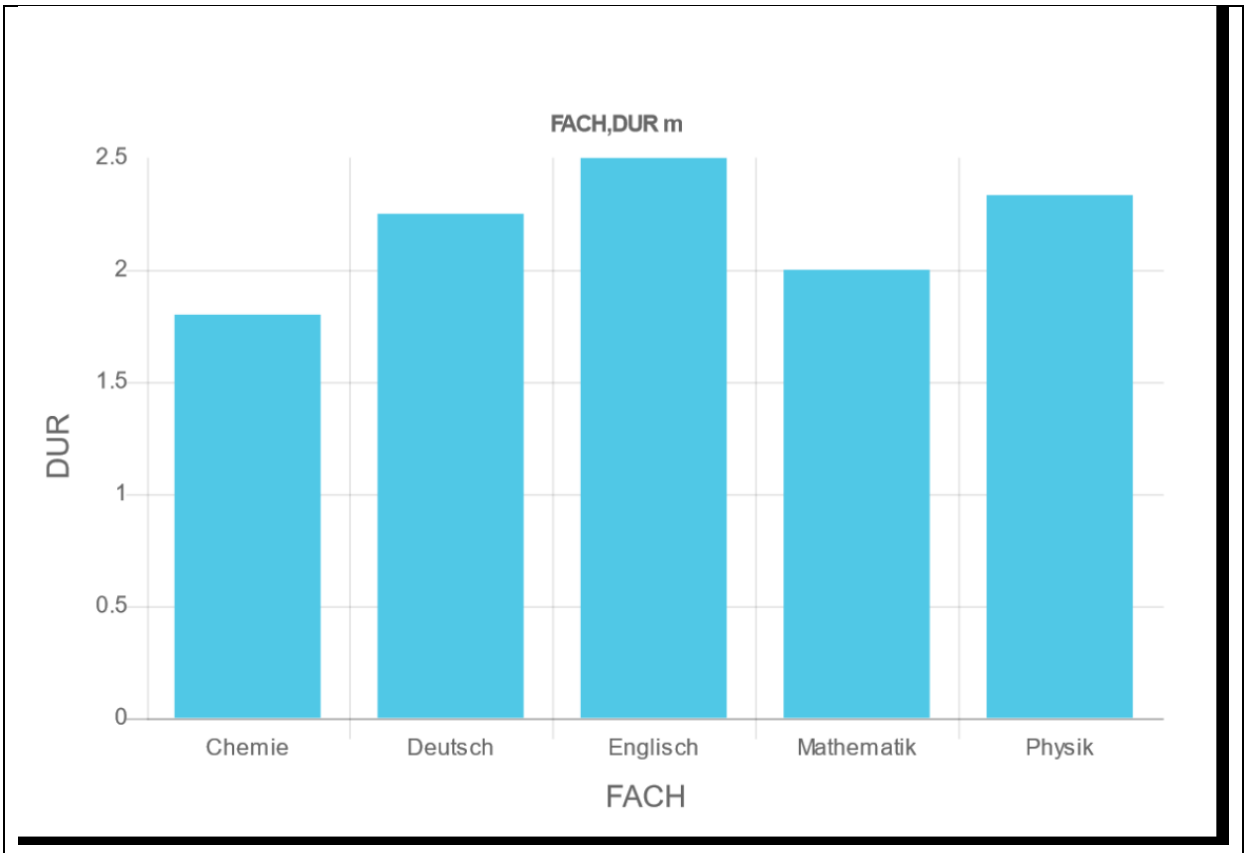
Programm 20: Operationen die Tupel (Paare bzw. Tripel) von Zahlen zurückgeben
`QUOTIENT,REST:=14 divrest 5          # Division mit Rest: binär` `TAG,MON,JAHR:=26.03.1967 zahltrip   # Zahlentripel : unär`
Ergebnis (tab)
`QUOTIENT, REST, TAG, MON, JAHR`
`2         4     26   3    1967`

3.2 Diagramme

Programm 21: ein einfaches Diagramm erzeugen

`meinenoten.tabh`
`gib FACH,DUR m    (# Zielschema für 2 spaltige Menge #)`
`    DUR:=NOTE! ++: # NOTEn werden zum Durchschnitt verarbeitet`

Ergebnis (Diagramm-Säulen)



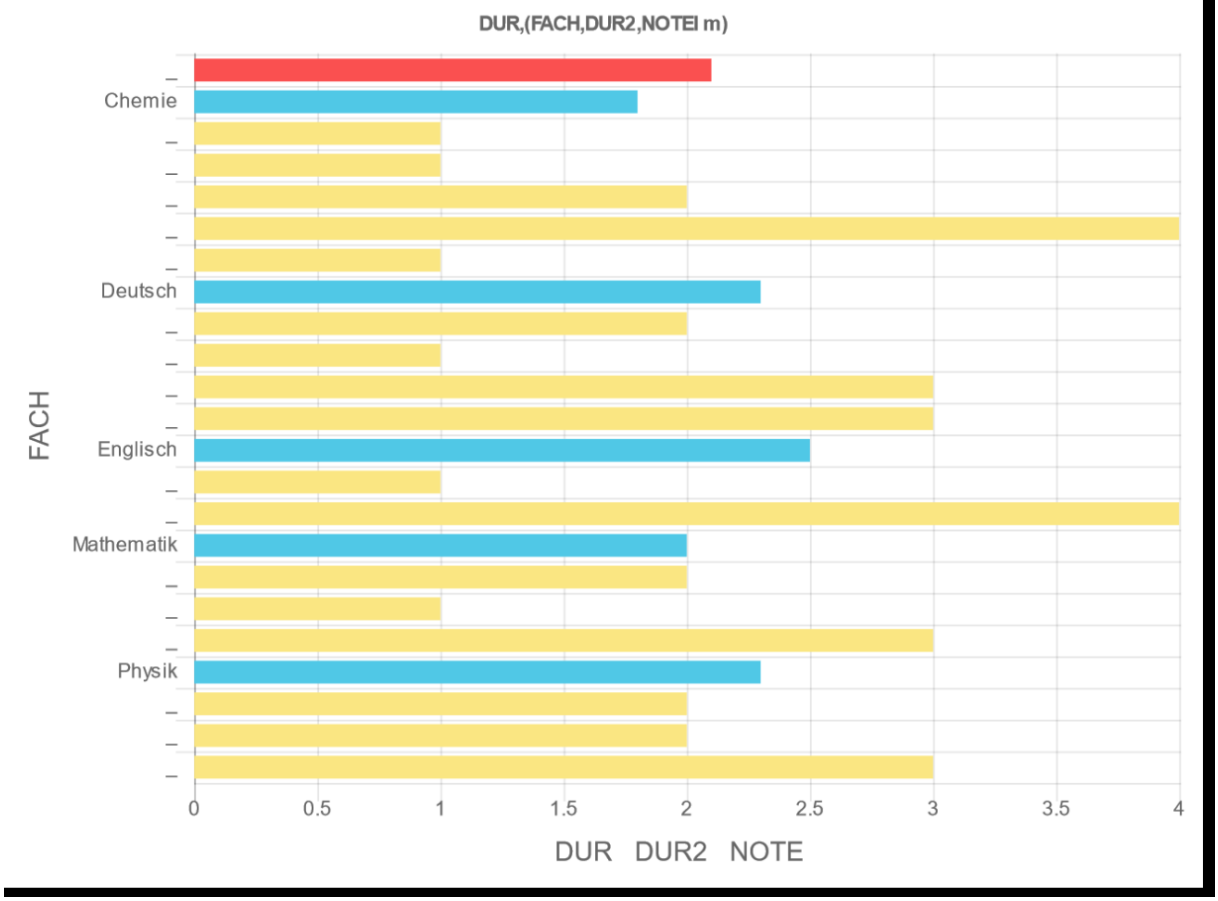
Ergebnis (tab-Ausgabe)

```
FACH,       DUR m
Chemie      1.8
Deutsch     2.25
Englisch    2.5
Mathematik  2.
Physik      2.33333333333
```

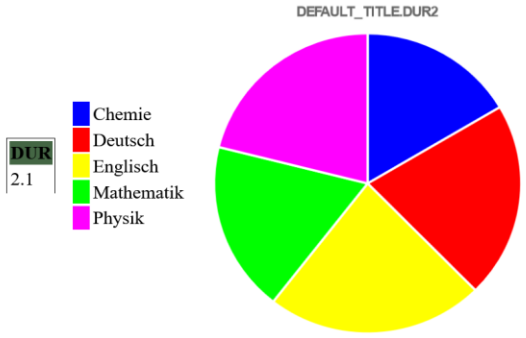
Programm 22: ein strukturiertes und ein multiples Diagramm erzeugen

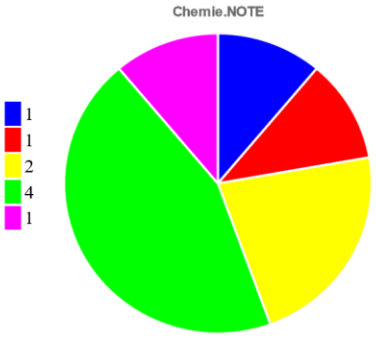
```
meinenoten.tabh
gib DUR,(FACH,DUR,NOTEl m)(# Schema mit 3 Ebenen; nullte Ebene für Ge- #)
    DUR:=NOTE! ++:          # samtdurchschnitt; erste Ebene für Durchschnitt
rnd 1                       # pro Fach
```

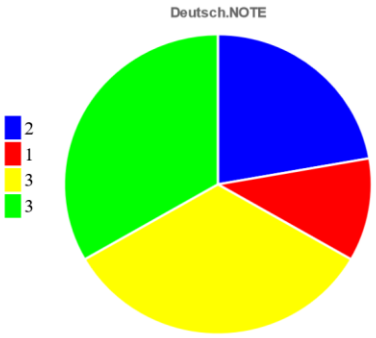
Ergebnis (strukturiertes Balkendiagramm)

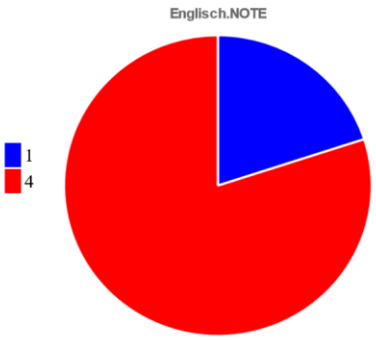


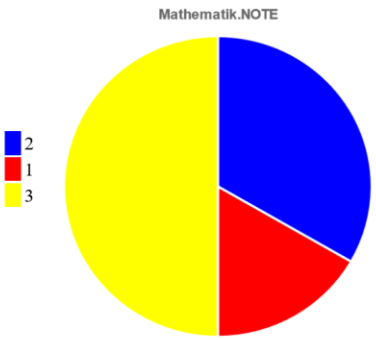
Ergebnis (multiples Kreisdiagramm)

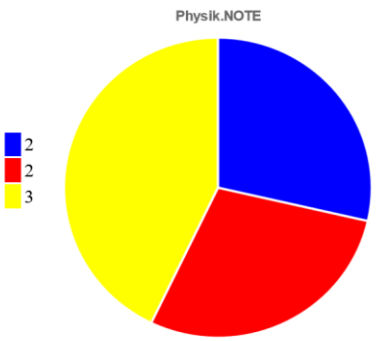

DEFAULT_TITLE.DUR2
DUR
2.1
Chemie
Deutsch
Englisch
Mathematik
Physik


Chemie.NOTE
1
1
2
4
1


Deutsch.NOTE
2
1
3
3


Englisch.NOTE
1
4


Mathematik.NOTE
2
1
3


Physik.NOTE
2
2
3

Ergebnis (tabh)

```
DUR, (FACH,      DUR2, NOTEl  m)
2.1   Chemie     1.8   1 1 2 4 1
      Deutsch    2.3   2 1 3 3
      Englisch   2.5   1 4
      Mathematik 2.0   2 1 3
      Physik     2.3   2 2 3
```

Damit die Legende der rechten 5 Kreise erscheint, wurde die folgende Zuweisung angehängt.

NAMENOTE:=NOTE wort

Die Spalte NAMENOTE ist fast mit der Spalte NOTE identisch. Sie unterscheidet sich lediglich durch den Typ. ZAHL wurde in WORT umgewandelt.

3.3 Schulprobleme

Programm 23: Finde auf einfache Weise das lokale Maximum der Sinusfunktion im Intervall [1, 3].

```
LOKALES_MAXIMUM:=1 ...3.!0.000'01 sin max
                   # ... ist 3-stellig
                   # der zweite und dritte Inputwert wird durch ! getrennt
```

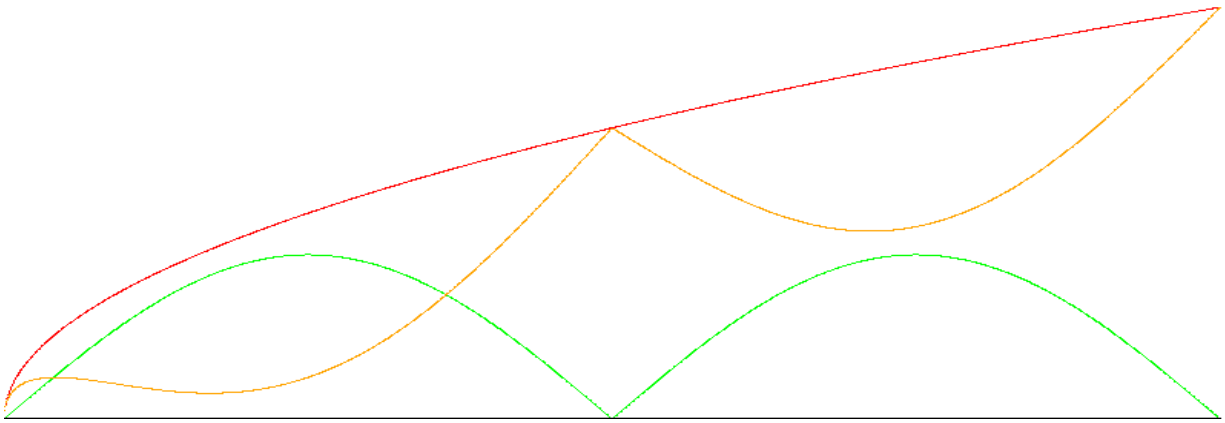
Ergebnis

```
LOKALES_MAXIMUM
0.999999999993
```

Programm 24: Plotte den Betrag der Sinusfunktion, die Quadratwurzel, die Differenz der beiden und die X-Achse im Intervall [0,2*pi].

```
Xl:= 0 ... 2*pi!0.003        # von ... bis ! schrittweite
SINUS    :=X sin abs         # auf jeden X- Wert wird zunächst die
WURZEL   :=X sqrt            # Sinusfunktion und dann der absolute Betrag
DIFF     :=WURZEL - SINUS    # angewandt
Y        :=X*0
RGBSINUS :=green  leftat SINUS
RGBWURZEL:=red    leftat WURZEL
RGBDIFF  :=orange leftat DIFF
RGBY     :=black  leftat Y
```

Ergebnis (bild):



Programm 25: Berechne die Fläche zwischen der obigen schwarzen Geraden und der grünen Funktion durch Summierung von vielen sehr kleinen Rechteckflächen (Algorithmus von Archimedes).

```
FLAECHE:=0 ... 2*pi!0.000'1 sin abs *0.000'1  ++
```

Ergebnis

```
FLAECHE
3.99999999798
```

3.4 Ein 4 zeiliges o++o-Programm, für das EXCEL mehr als sechs Arbeitsblätter benötigt!

Die Selektion dürfte eine der wichtigsten Operationen der Digitalisierung sein. Niemand will und kann eine Datenausgabe von einer Millionen Datensätzen durch Betrachten einer Tabelle verarbeiten. Meistens interessiert man sich nur für sehr wenige Daten, die dann selektiert werden müssen. Oder man wünscht aggregierte Daten.

Mit avec (mit) und sans (ohne) können vielfältige Bedingungen formuliert werden. Die Spaltennamen und die Struktur der Daten bleiben bei Selektionen erhalten.

```
NAME,     LAENGE, (ALTER, GEWICHT m)m
Klaus     1.68     18     61
                   30     65
                   61     80
Rolf      1.78     40     72
Kathi     1.70     18     55
                   40     70
Walleri   1.00      3     16
Viktoria  1.61     13     51
Bert      1.72     18     66
                   30     70
```

bmidaten.tab

Programm 26: Gib alle Personen aus, die größer als 1.70 m sind.

```
aus  bmidaten.tab
avec LAENGE>1.70
```

Ergebnis (tab)

```
NAME, LAENGE, (ALTER, GEWICHT m) m
Bert  1.72     18     66
               30     70
Rolf  1.78     40     72
```

Programm 27: Streiche alle Personen, die größer als 1.70 sind.

```
aus  bmidaten.tab
sans LAENGE>1.70
```

Ergebnis (tab)

```
NAME,     LAENGE, (ALTER, GEWICHT  m) m
Kathi     1.7      18     55
                   40     70
Klaus     1.68     18     61
                   30     65
                   61     80
```

```
Viktoria 1.61     13     51
Walleri  1.        3     16
```

Programm 28: Gib zu **jeder** Person alle seine Daten, mit einem Alter größer 20 aus.

```
aus  bmidaten.tab
avec ALTER! ALTER>20
```

Ergebnis (tab)

```
NAME,    LAENGE, (ALTER, GEWICHT  m) m
Bert     1.72     30     70
Kathi    1.7      40     70
Klaus    1.68     30     65
                  61     80
Rolf     1.78     40     72
Viktoria 1.61
Walleri  1.
```

Programm 29: Gib alle Erwachsenen aus. Genauer: Alle Personen mit all ihren Daten, die einen Alterseintrag größer als 20 besitzen.

```
aus  bmidaten.tab
avec NAME! ALTER>20
```

Ergebnis (tab)

```
NAME, LAENGE, (ALTER, GEWICHT  m) m
Bert  1.72     18     66
               30     70
Kathi 1.7      18     55
               40     70
Klaus 1.68     18     61
               30     65
               61     80
Rolf  1.78     40     72
```

Zu beachten ist hierbei, dass die Subtupel mit dem Alter 18 auch ausgegeben werden, da im entsprechenden Tupel ein (anderes) Subtupel vorkommt, das die Bedingung ALTER>20 erfüllt.

Programm 30: Strukturiere die BMI-Daten um, indem zu jedem Alter alle Namen mit Gewicht aufgelistet werden, für die dieser Gewichtseintrag existiert. Die Daten sind nach Alter und innerhalb einer (NAME,GEWICHT)-Kollektion nach NAME zu sortieren.

```
aus bmidaten.tab
gib ALTER,(NAME,GEWICHT  m) m
```

Ergebnis (tab)

```
ALTER, (NAME,    GEWICHT  m) m
 3      Walleri  16
13      Viktoria 51
18      Bert     66
        Kathi    55
        Klaus    61
30      Bert     70
        Klaus    65
40      Kathi    70
        Rolf     72
61      Klaus    80
```

Programm 31: Für alle Erwachsenen sind alle Body Maß Indexe zu bestimmen. Die gegebene Tabelle ist umzustrukturieren. Das Ergebnis ist nach ALTER und NAME zu sortieren.

```
aus  bmidaten.tab
avec NAME! ALTER>20
gib  BMI,(ALTER,BMI,(NAME,BMI m) m) BMI:=GEWICHT:LAENGE:LAENGE!++:
rnd  2
```

Ergebnis

```
BMI, (ALTER, BMI2, (NAME, BMI3  m) m)
23.12 18     20.98  Bert  22.31
                    Kathi 19.03
                    Klaus 21.61
      30     23.35  Bert  23.66
                    Klaus 23.03
      40     23.47  Kathi 24.22
                    Rolf  22.72
      61     28.34  Klaus 28.34
```

Die obige Bedingung NAME! ALTER>20 selektiert Personen, die einen ALTER-Eintrag grösser als 20 enthalten. Der hier involvierte Existenzquantor wird nicht geschrieben. Somit werden die Subtupel mit dem Alter 18 nicht verworfen, da noch höhere Alterseinträge vorhanden sind. Auf die Visualisierung der strukturierten Ergebnistabelle wird an dieser Stelle verzichtet.

3.5 Tabellenkalkulation mit o++o

Programm 32: Stelle das Wirtschaftswachstum zweier Länder mit durchschnittlich 1.1 % bzw. 8.9 % Wachstum gegenüber.

```
JAHRl:= 2010 .. 2020
L1:= first 100. next L1 pred +% 1.1 at JAHR # pred ist der Vorgänger
L2:= first 100. next L2 pred +% 8.9 at L1
rnd 1
```

Ergebnis (tab)

```
JAHR, L1,   L2  l
2010  100.0 100.0
2011  101.1 108.9
2012  102.2 118.6
2013  103.3 129.1
2014  104.5 140.6
2015  105.6 153.2
2016  106.8 166.8
2017  108.0 181.6
2018  109.1 197.8
2019  110.3 215.4
2020  111.6 234.6
```

Die L1 und L2-Werte der ersten Zeile werden durch die erste "Formel" (100.) errechnet. Alle weiteren Werte werden durch die zweite Formel jeweils aus den Vorgängerdaten berechnet.

Programm 33: Bestimme das Wachstum eines Bankkontos von 1'200 Euro bei 5 vorgegebenen Zinssätzen.

```
ZINSl:= 1.1 2.3 4.3 5.4 2.4
STAND:= first 1200 +% ZINS next STAND pred +% ZINS at ZINS
```

```
rnd 1
```

Ergebnis (tab)

```
ZINS, STAND  l
1.1   1213.2
2.3   1241.1
4.3   1294.5
5.4   1364.4
2.4   1397.1
```

Programm 34: Stelle die Entwicklung eines Darlehens von 15'000 Euro bei 165 Euro monatlicher Zahlung bei 5.9 % effektivem Zins dar. Es soll lediglich der Januarwert jeden Jahres ausgegeben werden. Die Berechnung soll monatsweise erfolgen.

```
MONZINS:=1.059 hoch 1/12 - 1 *100
MON,STAND l:= while STAND >= 165
              first 0,15'000.
              next MON pred +1,(STAND pred +% MONZINS - 165) at MONZINS
avec MON rest 12=0
JAHR:=MON div 12 +2023
STAND::= STAND rnd 1 '3          # ::= : Überschreiben
gib JAHR,STAND l
```

Ergebnis (tab)

```
JAHR, STAND  l
2023  15'000.0
2024  13'852.0
2025  12'636.3
2026  11'348.8
2027   9'985.4
2028   8'541.6
2029   7'012.6
2030   5'393.3
2031   3'678.5
2032   1'862.6
```

Programm 35: Erstelle für die Liquiditätsplanung eines Startups die Einnahmen für die ersten 12 Monate. Gehe für jedes deiner Produkte von einem realistischen exponentiellen Wachstum von einem bestimmten Zeitpunkt aus.

```
MONl     := 1 ..12
ANDROID  := 0. falls MON <= 2 ! (1.3  hoch MON - 2)  # falls ist 3-stellig
POSTGRES := 0. falls MON <= 7 ! (1.25 hoch (MON - 7))
ORACLE   := 0. falls MON <= 7 ! (1.2  hoch (MON - 7))
IOS      := 0. falls MON <= 2 ! (1.3  hoch MON * 0.4)
WINDOWS  := 0. falls MON <= 5 ! (1.4  hoch (MON - 4))
LINUX    := 0. falls MON <= 5 ! (1.2  hoch MON *0.2)
EINNAHMEN:= MON tup ++ - MON
TOTAL:=EINNAHMENl ++
rnd 2
```

Ergebnis (tab)

```
TOTAL ,(MON ,ANDROID ,POSTGRES ,ORACLE ,IOS ,WINDOWS ,LINUX ,EINNAHMEN  l)
184.74   1    0.00    0.00      0.00    0.00  0.00    0.00    0.00
```

```
 2    0.00    0.00      0.00    0.00  0.00    0.00    0.00
 3    0.20    0.00      0.00    0.88  0.00    0.00    1.08
 4    0.86    0.00      0.00    1.14  0.00    0.00    2.00
 5    1.71    0.00      0.00    1.49  0.00    0.00    3.20
 6    2.83    0.00      0.00    1.93  1.96    0.60    7.31
 7    4.27    0.00      0.00    2.51  2.74    0.72   10.25
 8    6.16    1.25      1.20    3.26  3.84    0.86   16.57
 9    8.60    1.56      1.44    4.24  5.38    1.03   22.26
10   11.79    1.95      1.73    5.51  7.53    1.24   29.75
11   15.92    2.44      2.07    7.17 10.54    1.49   39.63
12   21.30    3.05      2.49    9.32 14.76    1.78   52.70
```

Wenn die Zahlen für die Android App beispielsweise zu groß ausgefallen sind, könnte man die Basis 1.3 durch 1.25 ersetzen und die Berechnung durch einen Klick aktualisieren.

3.6 Strukturierte Dokumente

Programm 36: Gesucht sind die ersten 300 Buchstaben der Einführungen der Dokumente der deutschen Wikipedia der Städte Rom und Paris.

```
wiki                              # wiki: deutsche Miniwikipedia
keys Paris Rom                    # oder keys [Paris Rom]
avec ANR=0                        # ANR: Abschnittsnummer
INHALT::=INHALT subtext 1!300     # subtext ist dreistellig
gib TITEL,INHALT l
```

Ergebnis (web)

TITEL	INHALT
Paris	'''Paris''' ( ) ist die Hauptstadt der Französischen Republik und Hauptort der Region Île-de-France. Der Fluss Seine teilt die Stadt in einen nördlichen (Rive Droite, „rechtes Ufer“) und einen südlichen Teil (Rive Gauche, „linkes Ufer“); administrativ ist sie in 20 Stadtbezirke ''(Arrondissements)''
Rom	'''Rom''' (;), amtlich , ist die Hauptstadt Italiens[[Ref]]. Mit etwa drei Millionen Einwohnern im Stadtgebiet bzw. rund vier Millionen Einwohnern in der Agglomeration ist sie die größte Stadt Italiens. Rom liegt in der Region Latium an den Ufern des Flusses Tiber. Rom wurde erstmals im 1. Jahrhund

subtext ist eine dreistellige Textoperation:
text **subtext** Position des ersten Buchstaben maximale Länge des gewünschten Textes.

Programm 37: Gib das Inhaltsverzeichnis des Pariser Eintrags.

```
wiki
keys Paris           # Selektion des Eintrags mit dem ersten Wert "Paris"
gib  ANR,ATITEL l
```

Ergebnis

ANR,	ATITEL 1
0	Einleitung
1	Geografie
1.1	Lage
1.2	Klima
1.3	Geologie
1.4	Seine
1.5	Inseln
1.6	Hügel
1.7	Stadtgliederung
2	Geschichte
2.1	Antike
2.2	Mittelalter
2.3	Neuzeit
3	Hoheitssymbole
4	Gesellschaft
4.1	Demografie
4.2	Einwanderung
4.3	Religionen
5	Politik
5.1	Stadtregierung
5.2	Stadtrat (''Conseil de Paris'')
5.3	Städtepartnerschaften
6	Kultur und Sehenswürdigkeiten
6.1	Theater
6.2	Museen
6.3	Bauwerke
6.3.1	Brücken
6.3.2	Plätze und Straßen
6.3.3	Weltliche Bauwerke
6.3.3.1	Antike
6.3.3.2	Mittelalter
6.3.3.3	Frühe Neuzeit
6.3.3.4	17. Jahrhundert
6.3.3.5	18. Jahrhundert
6.3.3.6	19. Jahrhundert
6.3.3.7	20. Jahrhundert
6.3.3.8	21. Jahrhundert
6.3.4	Kirchen
6.3.4.1	Mittelalter
6.3.4.2	Neuzeit
6.4	Grünflächen
6.4.1	Promenaden, Parks und Gärten
6.4.2	Friedhöfe
6.5	Film
6.6	Sport
6.6.1	Sportveranstaltungen
6.6.2	Sportstätten
6.7	Regelmäßige Veranstaltungen
6.8	Gastronomie
6.9	Einkaufen
6.10	Sehenswürdigkeiten in der Umgebung
7	Wirtschaft und Infrastruktur

```
7.1     Wirtschaft
7.2     Verkehr
7.2.1   Fernverkehr
7.2.2   Nahverkehr
7.2.3   Luftqualität
7.3     Wissenschaft und Bildung
8       Persönlichkeiten
8.1     Ehrenbürger
8.2     Söhne und Töchter der Stadt
8.3     Persönlichkeiten, die vor Ort gewirkt haben
9       Siehe auch
10      Literatur
10.1    list_element
...
10.13   list_element
11      Einzelnachweise
```

Programm 38: Gib die Abschnittsnummern mit den Abschnittstiteln des Pariser Eintrags, die das Wort Berlin in der Spalte INHALT enthalten.

```
wiki
keys Paris
avec ANR! Berlin in INHALT
gib  ANR,ATITEL l
```

Ergebnis (tab)

```
ANR,  ATITEL l
4.1   Demografie
5.3   Städtepartnerschaften
7.1   Wirtschaft
10.10 list_element
```

Programm 39: Gib das Inhaltsverzeichnis von Kapitel 7.

```
wiki
keys Paris
avec ANR subtext 1!1=7
gib  ANR,ATITEL l
```

Ergebnis:

```
ANR,  ATITEL l
7     Wirtschaft und Infrastruktur
7.1   Wirtschaft
7.2   Verkehr
7.2.1 Fernverkehr
7.2.2 Nahverkehr
7.2.3 Luftqualität
7.3   Wissenschaft und Bildung
```

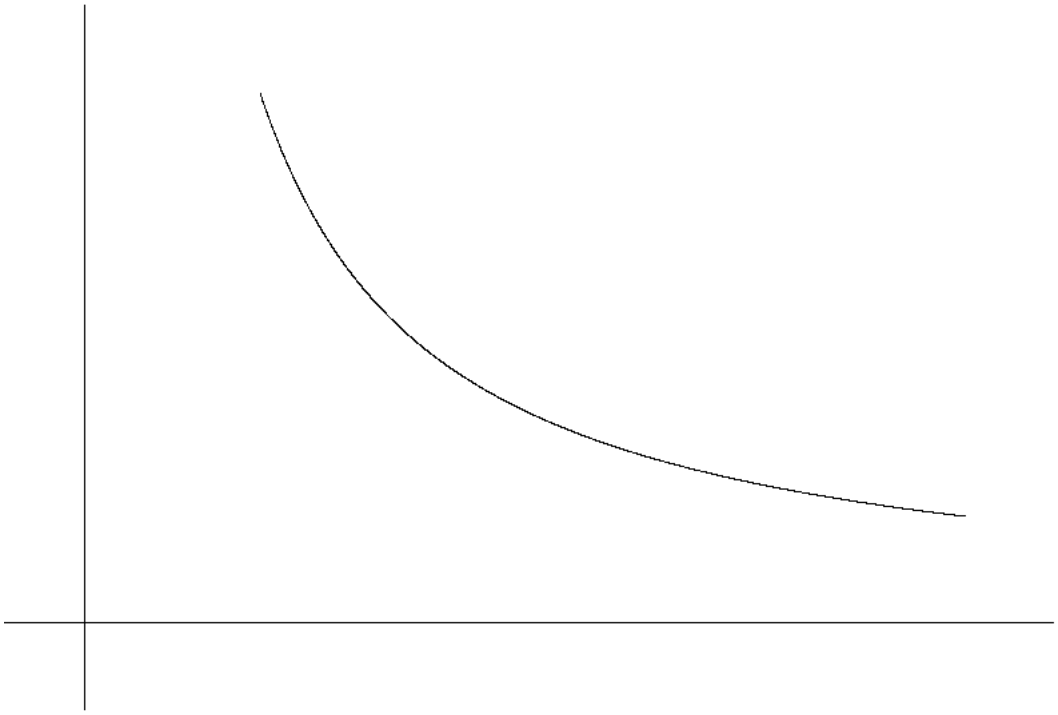
3.7 Ein Anfangswertproblem einer Differentialgleichung

Programm 40: Plotte die Lösung des Anfangswertproblems X0=2 Y0=6 für die Differentialgleichnung y'= -y : x mit Koordinatenachsen.

```
Xl:= 2 ... 10! 0.002 #at ANF
Y:=first 6 next Y pred + (((Y pred * (-1)) :X pred)*0.002) at X
```

```
X0l:=-1 ...11!0.002     # X-Achse
Y0:=X0*0                # X-Achse
Y1l:= -1 ...7!0.002     #Y-Achse
X1:=0 leftat Y1         #X-Achse
```

Ergebnis (bild):



4 Die Designprinzipien von o++o

Wir fassen die wichtigsten Gestaltungsprinzipien und Anforderungen an eine Endnutzer-Computersprache oder ein Datenmodell mit entsprechenden Operationen zusammen:

1. Es sollte auf leicht anwendbaren Konzepten mit einer einfachen Syntax basieren.
2. Es sollte ausdrucksstark und leistungsfähig sein.
3. Es sollte durch neue Operationen erweiterbar sein.
4. Es sollte eine präzise Semantik haben, die auf Algorithmen basiert.
5. Es sollte Anfragen auf Tabellen (Datenbanken) und Dokumente ermöglichen.
6. Es sollte Anfragen über Dokumentkollektionen (IR-Systeme) und ganze Datenbanken ermöglichen.
7. Es sollte Berechnungen durch naive (brute force) Algorithmen ermöglichen.
8. Es soll auch für Menschen mit wenig Interesse an Mathematik und Informatik nutzbar sein.
9. Es soll neben einfachen auch anspruchsvollere Konzepte für breite Anwendungsklassen bieten, die auch für Benutzer mit großem Interesse an Mathematik und Informatik geeignet sind.
10. Es sollte Einzeldaten- und Massendatenoperationen solide integrieren.
11. Es wäre schön, wenn es grafische Features auf Basis von strukturierten Tabellen nutzen könnte.
12. Es sollte effizient umsetzbar sein.

13. Zumindest der Kern der Sprache sollte optimiert werden können.

o++o wurde unter Berücksichtigung dieser Prinzipien konzipiert und entwickelt. Es begann als Datenbanksprache für Tabellen mit Wiederholgruppen. Ein Datensatz mit Wiederholunggruppen darf an jeder Position nicht nur einen Wert enthalten, sondern auch mehrere (Subtupel von) Werten. Beispielsweise darf ein Studentensatz einen Namen und ein Stipendium enthalten, aber auch mehrere Hobbies oder mehrere (FACH, NOTE-) Paare. Ebenso darf ein Maschinenteil eine Nummer und die Bezeichnung einer Farbe enthalten und zudem mehrere Unterteile oder mehrere Ebenen oder Kanten. Solche Subtupel dürfen wieder Subtupel enthalten. Diese Wiederholgruppen gibt es in der Informatik bereits seit mehr als 50 Jahren. Sie sind typisch für hierarchische Systeme (IMS,...), wurden aber später durch aufkommende relationale Systeme in Verruf gebracht. Auch heute sind sie in XML-, JSON- und NoSQL-Systemen weit verbreitet. Unserer Meinung nach gibt es jedoch keine allgemein akzeptierte Computersprache, die in der Lage ist, diese reichhaltigeren Strukturen angemessen zu handhaben. Mit dem Aufkommen von XML konnten wir unsere Operationen auf die neuen Möglichkeiten des beliebigen Tagging und des Alternate-operators (|) verallgemeinern. Daher sind wir in der Lage, nicht nur Tabellen, sondern auch Dokumente zu verarbeiten. Wir haben den Namen Tabment eingeführt. Ein Tabment kann im Wesentlichen als abstrakte (syntaxunabhängige) Präzisierung eines XML-Dokuments verstanden werden.

Schritt für Schritt haben wir unsere Sprache o++o verbessert. Wir haben binäre Suchbäume in Tabmente eingeführt. So haben wir für mehrere Operationen große Effizienzsteigerungen erzielt.

Indizes können auch als Tabmente betrachtet werden.

Unsere Sprache o++o ist in OCaml implementiert. Einige grundlegende Schlüsselwörter von o++o sind Deutsch oder Französisch ('gib' anstelle von SELECT, 'avec' anstelle von WHERE,...), weil sie kürzer sind als die entsprechenden englischen Wörter, aber die meisten Schlüsselwörter sind Englisch. Kurze Schlüsselworte scheinen wichtig zu sein, da Smartphones nur einen kleinen Bildschirm besitzen.

Was sind die spezifischeren Designprinzipien von o++o?

1. **Das Wichtigste zuerst**

1.1 Sortieren nach den ersten Attributen einer Kollektion

```
gib ABTEILUNG,CHEF,(NAME,ORT m) m
```

Hier ist eine strukturierte Tabelle beschrieben, die zu jeder Abteilung auch eine entsprechende Gruppe von Mitarbeitern enthält. Mengen (m) (und Multimengen) werden immer nach den ersten Spaltennamen sortiert. D.h., die äußere Menge wird nach ABTEILUNG sortiert und die innere Menge nach NAME und dann nach ORT, da der NAME nicht immer ein Schlüssel in einer Abteilung ist.

1.2 Zuerst geschrieben - zuerst berechnet:

`2+3*4` ergibt `20`

Hier hat ein Rechteck eine längere Seite, die aus zwei 2 und 3 Meter langen Teilabschnitten besteht. Die andere Seite ist 4 Meter lang. Die Fläche ergibt 20.

`3*4+2` ergibt `14`

Wenn ich zwei Rechtecke habe, eines mit den Seitenlängen 3 und 4 und eines mit der Fläche 2, kann ich zuerst 3*4 berechnen und dann 2 addieren, um den Flächeninhalt zu erhalten.

1.3 TT-Invarianz (TT=TabmentTyp)

Bei vielen Operationen wie Addition oder Multiplikation ist der Typ des Ergebnisses derselbe wie der Typ des ersten Eingabewerts.

```
<TABH!
FACH,    NOTEl m
Mathe    1 2 4 1
Phy      2 3 5 2 4
TABH>
*15/6
```

Hier wird eine ganze Tabelle im horizontalen tab-Format mit einer Zahl multipliziert. Das heißt, jede Zahl der Tabelle wird mit 15/6 multipliziert und die Wörter bleiben unverändert. Somit ergibt sich wieder eine Tabelle vom Typ FACH,NOTEl m. Die Umbenennung von NOTE in PUNKTE kann vom Benutzer vorgenommen werden. D.h., auch hier ist der erste Eingabewert wichtiger als der zweite.

2. Pragmatik und Methodik zuerst

Wir können eine mehrzeilige Semantik auch für einen einzelnen Term zulassen. Dann könnten wir

```
(23+45+67) * (1111+2222+3333+4444)
```

durch

```
23+45+67
*
111+2222+3333+4444
```

ersetzen. Dies kann schneller getippt werden und ist auch übersichtlicher, indem jedem Klammerpaar eine Zeile gewidmet wird. In o++o wird diese Notation weiter verkürzt zu

```
23+45+67
* 1111+2222+3333+4444
```

Dies geschieht nicht aus methodischen Gründen (bessere Lesbarkeit), sondern aus Pragmatismus. Diese Notation verschwendet die zusätzliche mittlere Zeile nicht. Im Vergleich zur ersten Notation müssen Sie anstelle von 4 Klammern nur einmal eine (größere) Returntaste benutzen.

3. Kurze einprägsame Schlüsselwörter

Kurze Programme erfordern kurze Schlüsselwörter und kurze Operationssymbole oder -namen. Wenn die Anzahl dieser Symbole jedoch zu groß wird, muss man auch vollständige Namen für Bezeichnungen zulassen, damit sie sich der Benutzer merken kann. Bei o++o gilt: Je wichtiger ein Symbol ist, d. h. je häufiger es verwendet wird, desto kürzer ist es. Diese Regel kann besser umgesetzt werden, indem auch nicht-englische Wörter zugelassen werden.

Sehr kurz sind + , * ,... m, l ... . Das ist sicherlich in Ordnung. Wir haben zudem viele englische Begriffe durch einprägsamere und kürzere Symbole ersetzt:

sum: ++
product: **
average: ++:
count: ++1
…

Aus Dankbarkeit gegenüber den OCaml-Entwicklern aus Paris haben wir 2 französische Wörter für die Selektion eingeführt:

avec (mit), sans (ohne)

Wo wir sehr kurze einprägsame bekannte Wörter in einer anderen Sprache als Englisch gefunden haben, ersetzen wir englische Begriffe durch kürzere aus anderen Sprachen, wenn diese Wörter vielen Menschen bekannt sind:

true: si (spanisch italienisch)
false: no

Aus der Übersetzung von SELECT-FROM-WHERE (gib-aus-mit) sind die deutschen Wörter
gib (select) für „gib mir“
und

aus (from)
geworden.

4. **Programme werden von oben nach unten und von links nach rechts ausgewertet**
   Programme mit Schleifen oder allgemeiner Rekursion sind ausdrucksstark und leistungsfähig, aber oft schwer zu lesen und zu verstehen. Von sequentiellen Programmen wird erwartet, dass sie nicht so ausdrucksstark sind. o++o wurde auch entwickelt, um das Gegenteil zu beweisen. Dies erfordert leistungsstarke und ausdrucksstarke Operationen.

   Beispiel: Ist 37 keine Primzahl?
   Zunächst werden alle Produkte bis 100 berechnet:

```
Xl:=2 ..50            # 49 Zahlen generieren
Yl:=2 ..10 at X       # für jeden X-Wert 9 Zahlen erzeugen
PRODUKT:=X*Y          # alle Produkte berechnen
avec PRODUKT <= 100   # auswählen der gewünschten Produkte
gib PRODUKTm          # Produkte sortieren und Duplikate
                      # eliminieren
ANTWORT:= 37 in PRODUKTm   # Ist 37 ein Produkt?
```

   # ist das Kommentarzeichen.

Die **Lesbarkeit** von Programmen und Tabmenten ist ein wichtiges Problem.
o++o berücksichtigt das wie folgt:

1. Programme können häufig kurz geschrieben werden.
   An obigem Programm zur Bestimmung aller Produkte kann man einige Konzepte von o++o gut erklären. Will man nur wissen, ob 37 eine Primzahl ist, so kann das wesentlich kürzer formuliert werden:

```
TEILERl:= 2 .. 19
gib ANTWORT   ANTWORT:=37 rest TEILER = 0 ! ||
```

   Das Programm ist sicher gut lesbar, wenn man verinnerlicht hat, dass || die Existenzaggregation ist und man diese genauso anwenden kann wie die ++ Aggregation.
2. Zahlen können auch im Schweizer Stil dargestellt werden (z.B.: 12'345'678)
3. Um mehr als 4 Leerzeichen eingerückte Zeilen gehören logisch zur vorhergehenden Zeile.
   Z.B.:

```
meine_noten.tab
gib DUR,(FACH,DUR m)
    DUR:=NOTE! ++:   # diese Zeile gehört vom logischen
                     # Standpunkt noch zur vorangehenden
rnd 1
```

4. eine strukturierte Tabelle mit dem Schema
   `ABTEILUNG, CHEF, (NAME, GEHALT l) m`
   enthält jede Abteilung und jeden Chef nur einmal. Dies reduziert nicht nur die Redundanz, sondern verbessert auch die Lesbarkeit im Vergleich zu äquivalenten flachen Tabellen.

Eine Programmiersprache ist leichter **erlernbar**, wenn sie möglichst wenige aber universelle Konzepte bzw. Operationen benutzt.

1. Eine breite Anwendung finden die Operationen, die mit den Schlüsselworten *gib, avec, sans* und := verbunden sind.
2. m, m-, b, b- -Kollektionen werden stets nach den ersten Spaltennamen sortiert. Das ist nicht nur bei *gib*, sondern auch bei Zuweisungen (:=) gewährleistet. In l bzw. l- Kollektionen können beliebige Reihenfolgen vorliegen.

5 Danksagung

Ich möchte den folgenden Informatikern für ihre wertvollen Beiträge zu unserem System o++o und früheren Systemen danken:

Wolfgang Reichstein für die erste Implementierung der Restrukturierungsoperation in einem Schritt in C für HSQ-Dateien,

Dmitri Schamschurko für die erste Implementierung des ersten Kerns des „gib-aus-mit“-Konstrukts in einem funktionalen Stil (Caml Light),

Martin Schnabel für die Konzeption und Umsetzung von Unterprogrammen und weiteren Features,

Andreas Hauptmann für die Verbesserung vieler Konzepte in Design und Effizienz, insbesondere für erste Anfrageoptimierungskonzepte.

Weiterer Dank gilt Mirko Otto für die Unterstützung des o++o-Projekts.

Stephan Schenkl hat die Konzepte zu den strukturierten Diagrammen realisiert.

Danke an Eicke Redweik für die Implementierung des Zugriffs auf ein relationales DBMS und auf die Wikipedia und an Jens Winter für die Implementierung einer ersten o++o-App für Android.