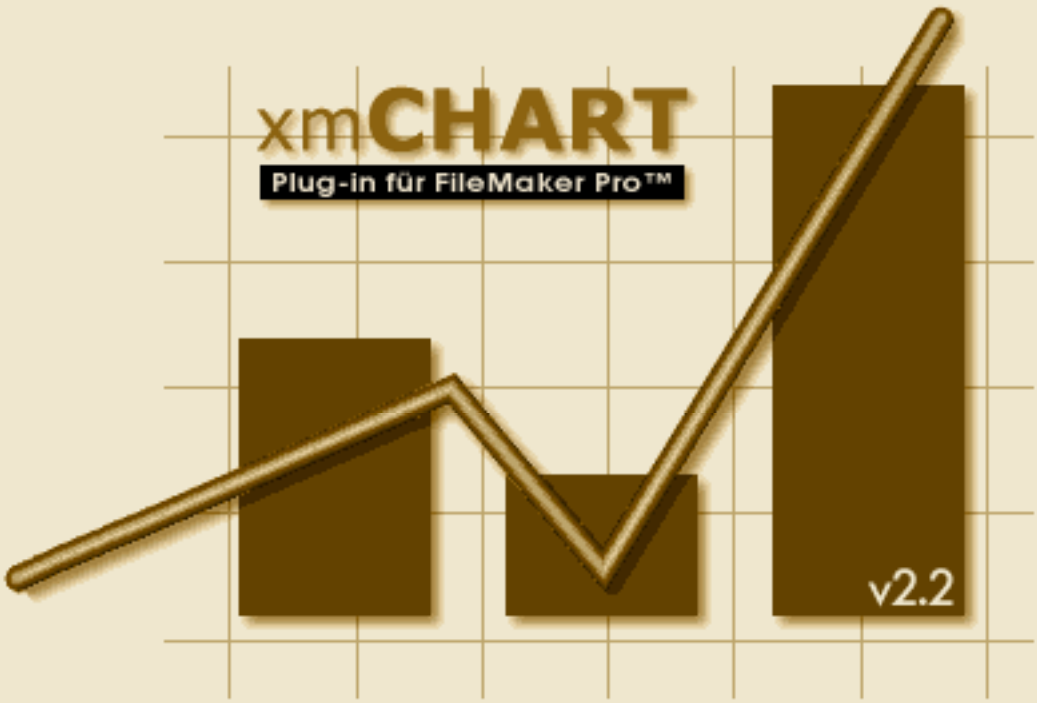
Tutorial



X2max

www.x2max.com

Inhaltsverzeichnis

Einführung

Komponenten

Einführung

Voraussetzungen

Betriebssystem:
•MacOS 8.6 oder höher
•Windows 95/98/Me/NT/2000/XP

Filemaker Pro Version:
Version: 4.0 oder höher

ACHTUNG:
Falls Sie noch FileMaker Pro Version 4.0v1 verwenden, entfernen Sie bitte alle nicht benötigten Plug-ins aus dem Ordner "FileMaker Erweiterungen" (unter MacOS/X) bzw. aus dem Ordner "System" (unter Windows), und stellen Sie sicher, dass die verbleibenden Plug-ins aktiviert sind. Ob ein Plug-in aktiviert ist, kann unter Menü *"Bearbeiten/Voreinstellungen/Programm..."* überprüft werden. Andernfalls werden zufolge eines Fehlers in Version 4.0v1, keine weiteren Plug-ins erkannt. Hilft das nicht, muß auf Version 4.0v2 oder höher aktualisiert werden. Dieser Fehler wurde ab FileMaker Pro Version 4.0v2 korrigiert.

FileMaker Pro Kenntnisse:
Um mit xmCHART erfolgreich arbeiten zu können, sind folgende FileMaker Pro-Kenntnisse notwendig:

•Definieren von FileMaker Pro Feldern, im speziellen, Variablen.
•Erstellen und Bearbeiten von Layouts.
•Handhabung von externen Funktionen. Über externe Funktionen werden Plug-ins in FileMaker Pro Datenbanken eingebunden.
•Grundlegende Kenntnisse in FileMaker Pro Scripting. Mittels Scripts kann die Erstellung von Diagrammen vollkommen automatisiert werden.

Alle angeführten Punkte werden ausführlich im *FileMaker Pro Benutzerhandbuch* erläutert. Darauf aufbauend wird in den nachfolgenden Abschnitten, auf sämtliche xmCHART betreffenden Kenntnisse und Besonderheiten detailliert eingegangen. Zusätzliche Ideen, Anregungen und Hilfestellungen finden sich in *xmBeispiele* inkl. mitgelieferter FileMaker Pro Beispieldateien.

Installation

Installation unter MacOS/X:

- Falls notwendig, beenden Sie FileMaker Pro.
- Kopieren Sie xmCHART (Plug-in) in den Ordner "FileMaker Erweiterungen", welcher sich normalerweise im "FileMaker Pro"-Ordner befindet.
- Starten Sie FileMaker Pro.

Weiters muß sichergestellt sein, dass xmCHART aktiviert ist. Dies kann unter Menü *"Bearbeiten/Voreinstellungen/Programm..."* überprüft werden. Unter FileMaker Pro Version 4.0v1 kann es, speziell wenn sie mehrere Plug-ins benutzen, zu Konflikten kommen. Abhilfe: siehe *Voraussetzungen*.

xmCHART kann auch problemlos in FileMaker Pro Runtime Lösungen eingebunden werden. Zu diesem Zweck erstellen Sie innerhalb des Ordners, welcher die FileMaker Pro Dateien für Ihre Runtime Lösung enthält, einen Ordner mit dem Namen "FileMaker Erweiterungen" und kopieren xmCHART (Plug-in für MacOS/X) in diesen Ordner.

Installation unter Windows 95/98/Me/NT/2000/XP:

- Falls notwendig, beenden Sie FileMaker Pro.
- Kopieren Sie xmCHART (Plug-in) in den Ordner "System", welcher sich normalerweise im "FileMaker Pro"-Ordner befindet.
- Starten Sie FileMaker Pro.

Weiters muß sichergestellt sein, dass xmCHART aktiviert ist. Dies kann unter Menü *"Bearbeiten/Voreinstellungen/Programm..."* überprüft werden. Unter FileMaker Pro Version 4.0v1 kann es, speziell wenn sie mehrere Plug-ins benutzen, zu Konflikten kommen. Abhilfe: siehe *Voraussetzungen*.

xmCHART kann auch problemlos in FileMaker Pro Runtime Lösungen eingebunden werden. Zu diesem Zweck erstellen Sie innerhalb des Ordners, welcher die FileMaker Pro Dateien für Ihre Runtime Lösung

enthält, einen Ordner mit dem Namen "System" und kopieren xmCHART (Plug-in für Windows) in diesen Ordner.

Wichtig:
Um die Druckqualität unter Windows OS zu verbessern, werden Zeichnungen standardmäßig im sog. Metafileformat erstellt und nicht im Bitmapformat. Um Diagramme im Metafileformat in einem FileMaker Pro Medienfeld verzerrungsfrei in der richtigen Größe darzustellen, sind die folgenden zwei Punkte zu beachten:

(1) Die Größe des Medienfeldes und die Größe des Diagramms müssen identisch sein. Die Breite und die Höhe des Diagramms werden durch die Funktion `OpenDrawing(breite;höhe)` festgelegt (in Pixel). Die Breite und Höhe des Medienfeldes können im Layout Modus unter dem Menüpunkt *Objektinfo...* kontrolliert werden. (Durch Mausklick auf eine der Dimensionsangaben [cm, px, in] am rechten Rand des Objektinfo-Fensters kann zwischen Zentimeter, Pixel und Inch hin- und hergeschaltet werden)

(2) Im Layout Modus ist unter dem Menüpunkt *Graphik...*, "Verkleinern/Vergrößern" einzustellen, und "Proportionen beibehalten" zu deaktivieren.

Support:
Sollten Probleme bei der Installation auftreten, bitte senden Sie eine entsprechende e-mail mit den folgenden Informationen:
•Plattform und Betriebssystemversion
•FileMaker Pro Version
•Fehlerbeschreibung
an: <support@x2max.com>. Wir bemühen uns, möglichst rasch mit Ihnen Kontakt aufzunehmen und das Problem zu lösen.

Grundlagen

Was ist xmCHART

xmCHART ist ein sehr mächtiges FileMaker Pro Plug-in zur grafischen Darstellung numerischer Daten. Dabei werden alle wichtigen Diagrammarten, wie Balken-, Punkte-, Linien-, Flächen-, HochTief- und Kuchendiagramme unterstützt. Zusätzlich werden auch statististische und finanztechnische Diagramme sowie Polar-, Radar und Ganttdiagramme zur Verfügung gestellt. Mittels eines umfangreichen Funktionensatzes können alle wichtigen Diagrammkomponenten wie etwa Skalierungen, Achsen, Raster oder Beschriftungen vom Benutzer variiert werden. Des weiteren können zur Vervollständigung eines Diagramms, Titel, Legende, Hintergrundbild, ergänzende Texte oder Bildinformationen, wie etwa ein Logo, einfach hinzugefügt werden.
xmCHART erschließt somit, sowohl dem FileMaker Pro Entwickler als auch dem Anwender eine Vielzahl neuer Möglichkeiten zur Auswertung und Darstellung numerischer Daten.

Wie funktioniert xmCHART

Die xmCHART zugrundeliegende Idee besteht darin, dem Anwender einen umfangreichen Satz an Funktionen zur Verfügung zu stellen, mit welchem das Erscheinungsbild eines Diagramms detailliert kontrolliert werden kann. Dabei reichen bereits drei Funktionen aus, um eine Vielzahl unterschiedlicher Diagramme zu erstellen; mit der ersten Funktion `OpenDrawing()` wird die Größe des Diagramms festgelegt, die zweite Funktion `ChartData()` enthält die darzustellenden Daten und die dritte Funktion legt die Art des Diagramms fest. Zum Beispiel:

```
OpenDrawing(400;300) // Breite & Höhe
ChartData(23 45 -2,76 12.9 5; 24 13 8.5 -3)
BarChart(shadow+horizontal)
```

Tatsächlich stehen aber derzeit über 140 Funktionen zur Verfügung, mit welchen beinahe alle Diagrammteile variiert werden können. Sämtliche Funktionsaufrufe werden entweder manuell oder automatisch mittels eines FileMaker Pro Scripts, zeilenweise, das heißt eine Funktion pro Zeile, in ein FileMaker Pro Textfeld eingetragen. Die Vorteile dieser Vorgangsweise sind:

- **Übersichtlichkeit**
  Alle zur Erstellung eines Diagramms benötigten Funktionsaufrufe sind übersichtlich, in einem einzigen Textfeld zusammengefaßt und können noch zusätzlich durch erklärende Kommentare ergänzt werden.

- **Einfaches Entwickeln und Testen**
  Änderungen können unmittelbar — ohne Aufruf des Script Editors — ausprobiert werden. Eventuell auftretende Fehler, werden durch Angabe der Zeilennummer, des Funktionsnamens und des Argumentindex genau spezifiziert.

  TIPP: *Nicht benötigte Funktionen können im Entwicklungs- und Teststadium einfach durch Vorsetzen eines Kommentarzeichens "//" deaktiviert werden.*

- **Portierbarkeit und Support**
  Funktionsaufrufe können einfach mittels der üblichen Editierbefehle *Kopieren* und *Einsetzen* in andere FileMaker Pro Datenbanken übernommen werden oder auch per e-mail versendet werden, zum Beispiel bei Fragen an den technischen Support:
  <support@x2max.com>

Soll nun ein Diagramm erstellt werden, wird der Inhalt dieses Textfeldes, welches im allgemeinen aus einer Anzahl von Funktionsaufrufen, Kommentarzeilen und Leerzeilen besteht, mittels eines kurzen FileMaker Pro Scripts an xmCHART übergeben. xmCHART überprüft anschließend diesen Textstring auf mögliche Tippfehler, fehlende Klammern, ungültige Werte, etc. Im Falle eines Fehlers wird abgebrochen und von xmCHART an FileMaker Pro eine Fehlermitteilung geliefert. Um dem Anwender ein rasches und einfaches Korrigieren zu ermöglichen, liefert xmCHART genaue Fehlermeldungen, mit Zeilennummer, und wenn möglich, mit Funktionsnamen und Argumentindex.
Werden keine Fehler gefunden, beginnt xmCHART mit dem Aufbau des Diagramms; dies geschieht für den Anwender unsichtbar im Hintergrund. Die Reihenfolge der Funktionsaufrufe hat in der Regel weder einen Einfluss auf die Geschwindigkeit noch auf das Aussehen des Diagramms.

Eventuell im Zuge des Diagrammaufbaus auftretende Fehler, wie etwa eine zu klein vorgegebene Diagrammgröße, führen ebenfalls zum Abbruch und zu einer entsprechenden Fehlermeldung. Tritt kein Fehler auf, wird abschließend das aufgebaute Diagramm von xmCHART in die Zwischenablage kopiert, von wo es mittels eines FileMaker Pro Scripts in ein Medienfeld kopiert werden kann.
Die Funktionsweise von xmCHART läßt sich also zusammengefaßt in folgende drei Bereiche gliedern:
•Überprüfung der eingegebenen Funktionsaufrufe.
•Aufbau des Diagramms *offscreen*, d.h. für den Anwender unsichtbar im Hintergrund.
•Kopieren des erstellten Diagramms in die Zwischenablage.

Ein erstes Beispiel

Im folgenden, einfachen Beispiel werden fünf Zahlenwerte durch ein Balkendiagramm grafisch dargestellt. Dabei sollen die Balken beschriftet und mit Schatten ausgeführt werden, weiters soll ein Diagrammtitel hinzugefügt werden. Die dafür notwendigen Schritte können in drei Aufgabenbereiche gegliedert werden.

- **FileMaker Pro Felder definieren**
  Im allgemeinen benötigt xmCHART drei FileMaker Pro Felder; zweckmäßigerweise sind das Variablenfelder, welche im folgenden durch das Prefix "g" (global) gekennzeichnet werden.

  Erstens, ein Textfeld, welches alle Funktionsaufrufe und optionale Kommentarzeilen enthält, im folgenden als *gFunktionen* bezeichnet.
  Zweitens, ein Medienfeld, welches das von xmCHART erstellte Diagramm übernimmt, im folgenden als *gDiagramm* bezeichnet.
  Und drittens, ein weiteres Textfeld *gFehler*, in welches eventuell auftretende Fehlermeldungen eingetragen werden. (Abb. 1)

Feldname	Feldtyp	Optionen
gFunktionen	Variable	Text
gDiagramm	Variable	Medien
gFehler	Variable	Text

Abb. 1

- **FileMaker Pro Scripts erstellen**

  In der Regel ist ein kurzes FileMaker Pro Script notwendig, welches einerseits den Inhalt von *gFunktionen* mittel der externen Funktion `xmCH-DrawChart()` an xmCHART übergibt und andererseits anschließend, das in der Zwischenablage gespeicherte Diagramm in das Medienfeld *gDiagramm* kopiert. Zusätzlich kann noch zu Beginn des Scripts abgefragt werden, ob die aktuelle FileMaker Pro Version Plug-ins unterstützt, das heißt, die FileMaker Pro Version muß 4.0 oder höher sein, und anschließend noch mittels der externen Funktion `xmCH-IsActive()`, ob xmCHART aktiv ist. (Abb. 2)

```
Script ausführen [Teilscripts, '_IstFMVersion4.0+']
Script ausführen [Teilscripts, '_IstPluginAktiv']
Feld angeben ['gFehler', 'Extern("xmCH-DrawChart"; gFunktionen)']
Einsetzen ['gDiagramm']
```

Abb. 2

Im Falle eines Fehlers wird in *gFehler* eine entsprechende Mitteilung gespeichert; tritt kein Fehler auf, so wird *gFehler* ein leerer Text ("") zugewiesen.
Die Funktionsweise der beiden Scripts *_IstFMVersion4.0+* und *_IstPluginAktiv* kann anhand der FileMaker Pro Beispieldateien, welche *xmBeispiele* beigefügt sind, studiert werden.

Weitere Scripts sind notwendig, wenn xmCHART-Funktionen dynamisch aufgebaut werden sollen, d.h. der Inhalt des Textfeldes *gFunktionen* wird mittels Scriptbefehle automatisch erstellt. Dies ist üblicherweise zur Erstellung der Diagrammdaten notwendig. Auf diese äußerst vielseitige und mächtige Handhabung von xmCHART wird in *xmBeispiele* ausführlich eingegangen.

- **xmCHART Funktionen**

  Die zum Aufbau und zur Gestaltung eines Diagramms benötigten Funktionen werden in das Textfeld *gFunktionen* eingetragen. Dabei sind drei Funktionsaufrufe unbedingt erforderlich, nämlich erstens die Funktion `OpenDrawing()`, welche die Größe des Diagramms festlegt, zweitens `ChartData()`, welche alle zur Darstellung verwendeten numerischen Werte enthält und drittens eine Funktion, welche die Art des Diagramms festlegt, z.B. `BarChart()`. Zur Zeit stehen folgenden Diagrammarten zur Verfügung: Balken-, Punkte-, Linien-, Flächen-, HochTief- und Kuchendiagramme, sowie statististische Diagramme (Histogramme, Box-Plots) und finanztechnische Diagramme (Candlestick-Diagramme, High-Low-

Close-Open-Diagramme), weiters Polar-, Radar und Ganttdiagramme.

Bei der Eingabe der Funktionen sind folgende Regeln zu beachten:

- Pro Zeile nur ein Funktionsaufruf; Leerzeilen sind erlaubt.
- Kommentare werden durch Voranstellen zweier Schrägstriche "//" gekennzeichnet.
- Besitzt eine Funktion mehrere Argumente, so werden diese durch Strichpunkte (";") getrennt.
- Optionale, d.h. nicht unbedingt erforderliche Argumente, können bei der Eingabe übersprungen werden. In diesem Fall werden die in xmCHART gespeicherten Standardwerte verwendet. Diese können aus *xmReferenz* entnommen werden. Zum Beispiel:
  `LegendBackground(white;;2;;;3)`
- Texte und Namen von Schriften, z.B. "Times" sind unter Hochkomma (") zu setzen. Soll ein Hochkomma ausgegeben werden, so ist dieses *doppelt* einzugeben. Zum Beispiel:
  `TitleText("""A""BC""")` bewirkt die Ausgabe: "A"BC"

ACHTUNG: *Hochkommas (") sind nicht mit typografischen Anführungszeichen (”“) zu verwechseln. Typografische Anführungszeichen können unter dem Menü "Bearbeiten/Voreinstellungen/Dokument..." aktiviert bzw. deaktiviert werden.*

Zur Erstellung des eingangs beschriebenen Diagramms sind die folgenden vier Funktionen erforderlichen: (Abb. 3)

```
OpenDrawing(250;200)
ChartData(8 12 10 5 7)
BarChart(label+shadow)
TitleText("Diagramm 1")
```

Diagramm 1

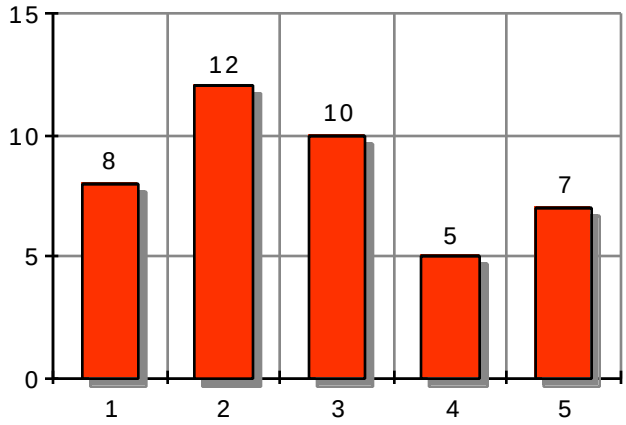


Abb. 3

Dazu noch folgende Anmerkungen:

- `OpenDrawing(breite;höhe)`
  `breite`: gesamte Breite des Diagramms inkl. Achsen, Legende, Titel, etc. (in Pixels)
  `höhe`: Die gesamte Höhe des Diagramms inkl. Achsen, Legende, Titel, etc. (in Pixels)

- `ChartData()`
  Die zur Darstellung kommenden numerischen Werte werden mittels der Funktion `ChartData()` an xmCHART übergeben. Die einzelnen Zahlenwerte einer Serie werden durch Leerzeichen getrennt. Bei der Übergabe mehrerer Serien sind diese durch Strichpunkte zu trennen; dabei kann die Anzahl der Werte pro Serie unterschiedlich sein. Zum Beispiel:

  ```
  ChartData(78 -12; 45 7 -23; 0; 12 -34 78 23,5)
  ```

- `BarChart()`
  Durch Eingabe sog. *Gestaltungskonstanten* kann das Aussehen eines Diagramms variiert werden. Die Gestaltungskonstanten können durch ein Pluszeichen "+" kombiniert werden. Welche Konstanten für welche Diagrammart zur Verfügung stehen, kann *xmReferenz* entnommen werden.

Komponenten

Komponenten

Übersicht

Ein mit xmCHART erstelltes Diagramm besteht in der Regel aus einer Vielzahl von Einzelkomponenten, wie z.B. Achsen, Raster, Legende, Titel oder dem Diagramm selbst. Nachfolgend werden alle in xmCHART zur Verfügung stehenden Funktionen, gruppiert nach Komponenten, zusammen mit zahlreichen Beispielen erklärt:

- Layout
- Daten
- Diagramme
- Stile
- Hintergrund
- Raster
- Achsen
- Legende
- Titel
- Hilfslinien
- Gleitende Durchschnitte
- Kurvenanpassung
- Fehlerbalken
- Grafische Basisfunktionen
- Ausgabe
- Diverses

Layout

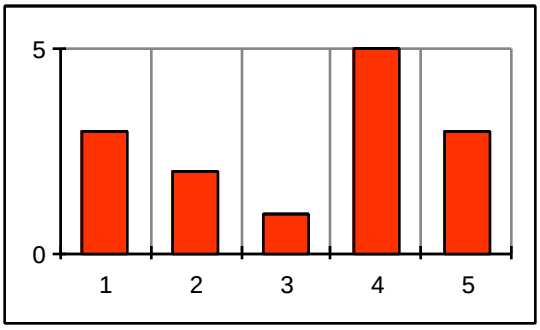
Insgesamt stehen sechs Layout-Funktionen zur Verfügung. Alle Funktionen treten paarweise in Form von *Open*- und *Close*-Funktionen auf. Sie dienen zum:

- Festlegen der Zeichnungsgröße.
- Präzisen Positionieren von Diagrammen.
- Überlagern von Diagrammen.
- Strukturieren komplexer Zeichnungen und Diagramme.

OpenDrawing(breite;höhe;typ)

Die wichtigste Layout-Funktion ist `OpenDrawing()`. Diese definiert die Breite und Höhe einer Zeichnung (in Pixels) und ist stets als erster Funktionsaufruf anzuführen. Zum Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
```



Optional kann als 3. Argument noch festgelegt werden, ob eine Zeichnung im Vektorformat (*typ=0*) oder im Bitmapformat (*typ=1*) erstellt werden soll. Fehlt das 3. Argument, so wird standardmäßig die Zeichnung im Vektorformat erstellt. Das heißt, unter MacOS/X im PICT-Format und unter Windows im EMF-Format. Das Vektorformat garantiert hohe Druckqualität, das Bitmapformat erweist sich zweckmäßig als plattformübergreifendes Format. Um Zeichnungen im EMF-Format (Windows) in einem FileMaker Pro Medienfeld richtig darzustellen, ist es notwendig, im Layout Modus, Menüpunkt *Graphik...*, "Verkleinern/Vergrößern" einzustellen, und

"Proportionen beibehalten" zu deaktivieren. Beispiel für Vektorformat:

```
OpenDrawing(200;100;0) // Vektorformat (default)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
```

Beispiel für Bitmapformat:

```
OpenDrawing(200;100;1) // Bitmapformat
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
```

Der Unterschied zwischen Vektor- und Bitmapformat kann durch Vergrößern der Zeichnung überprüft werden. Dies kann einfach durch den bei FileMaker Pro links unten platzierten Zoom-Button erfolgen.

CloseDrawing()

`CloseDrawing()` besitzt keine Argumente und kann optional als letzter Funktionsaufruf angeführt werden.
Ausgabefunktionen, wie zum Beispiel `SaveAsJPGFile()` können auch außerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()` angeführt werden.

OpenView(links;oben;breite;höhe)

Views ermöglichen die Unterteilung einer Zeichnung in mehrere Teilbereiche. So kann z.B. eine Serie von Grafikbefehlen, wie Linien, Flächen oder Texte innerhalb eines Views zusammengefaßt werden; durch einfaches Verändern der Position des Views, werden alle Grafikkomponenten innerhalb des Views automatisch mitverschoben. Andernfalls müßten die Koordinaten jedes einzelnen Grafikbefehls geändert werden. Das heißt, alle Koordinatenangaben innerhalb eines Views beziehen sich immer auf den View-Ursprung Links/Oben. Zum Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenView(80;40;100;70)
          AddFrame(0;0;100;70)
          ChartData(3 2 1 5 3)
          BarChart()
     CloseView()
CloseDrawing()
```

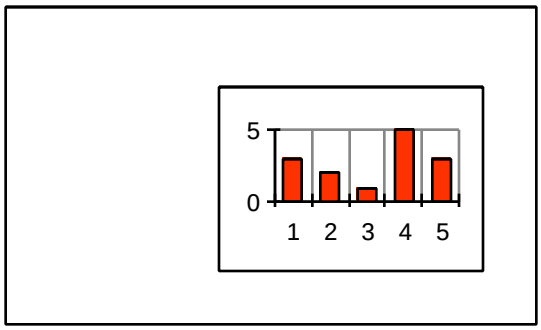
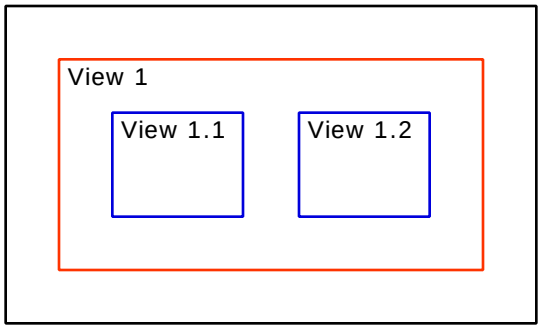


Diagramme und Grafikelemente, welche teilweise oder ganz außerhalb eines Views platziert sind, werden abgeschnitten (*geclippt*). Verschachtelte Views, d.h. Views innerhalb eines Views sind ebenfalls möglich.

Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenView(20;20;160;80)
          AddFrame(0;0;160;80;1;red)
          AddText(3;10;"View 1")
          OpenView(20;20;50;40)
               AddFrame(0;0;50;40;1;blue)
               AddText(3;10;"View 1.1")
          CloseView()
          OpenView(90;20;50;40)
               AddFrame(0;0;50;40;1;blue)
               AddText(3;10;"View 1.2")
          CloseView()
     CloseView()
CloseDrawing()
```



CloseView()

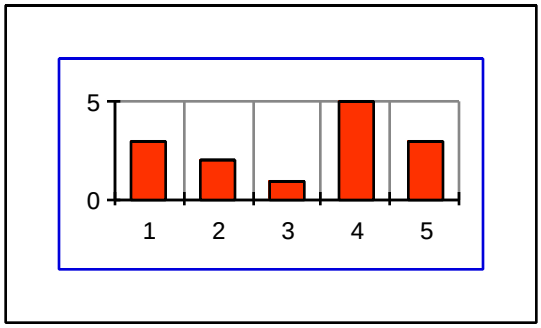
`CloseView()` besitzt keine Argumente und schließt die Befehlsfolge innerhalb eines Views ab. Wird nur ein einziger View definiert, kann auf `CloseView()` verzichtet werden.

OpenChart(links;oben;breite;höhe;istDiagrammfläche)

`OpenChart()` ermöglicht das präzise Positionieren eines Diagramms innerhalb einer Zeichnung oder eines Views. Dabei kann optional durch das fünfte Argument *istDiagrammfläche* entweder die gesamte — alle Diagrammkomponenten inkl. Achsenbeschriftungen, Titel und Legende umfaßende — Fläche festgelegt werden (*istDiagrammfläche=off*) oder nur die eigentliche Diagrammfläche, d.h. die vom Raster umschlossene Fläche (*istDiagrammfläche=on*).

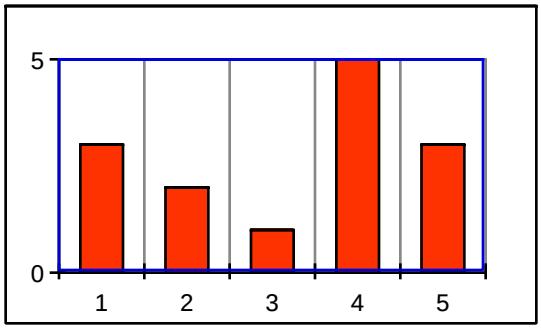
Beispiel für *istDiagrammfläche=off*: (default)

```
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    OpenChart(20;20;160;80;off)
        ChartData(3 2 1 5 3)
        BarChart()
        AddFrame(20;20;160;80;1;blue)
    CloseChart()
CloseDrawing()
```



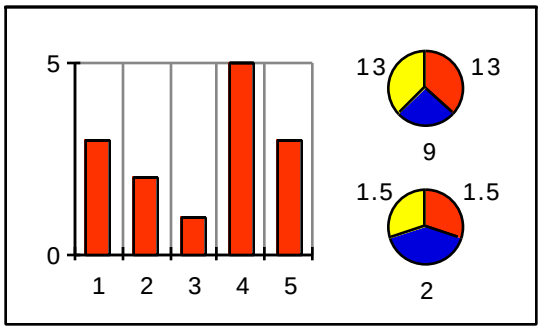
Beispiel für *istDiagrammfläche=on*:

```
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    OpenChart(20;20;160;80;on)
        ChartData(3 2 1 5 3)
        BarChart()
        AddFrame(20;20;160;80;1;blue)
    CloseChart()
CloseDrawing()
```



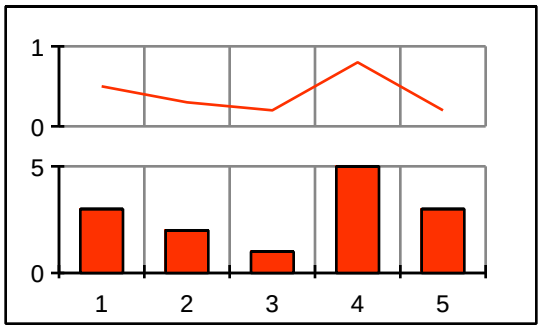
Das Argument *istDiagrammfläche=off* kann zur Platzierung mehrerer Diagramme innerhalb einer Zeichnung verwendet werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(5;5;120;115;off)
          ChartData(3 2 1 5 3)
          BarChart()
     CloseChart()
     OpenChart(120;0;75;75;off)
          ChartData(13 9 13)
          PieChart(label)
     CloseChart()
     OpenChart(120;50;75;75;off)
          ChartData(1.5 2 1.5)
          PieChart(label)
     CloseChart()
CloseDrawing()
```

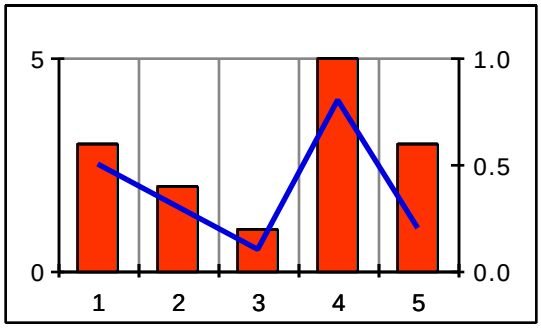


Das Argument *istDiagrammfläche=on* kann zum präzisen Aneinanderreihen oder Überlagern mehrerer Diagramme verwendet werden.
Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(20;15;160;30;on)
          ChartData(0.5 0.3 0.2 0.8 0.2)
          LineChart(;on)
          AxisOptions(x;none) // x-Achse ausblenden
     CloseChart()
     OpenChart(20;60;160;40;on)
          ChartData(3 2 1 5 3)
          BarChart()
     CloseChart()
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(20;20;150;80;on)
          ChartData(3 2 1 5 3)
          BarChart()
     CloseChart()
     OpenChart(20;20;150;80;on)
          ChartData(0.5 0.3 0.1 0.8 0.2)
          LineChart(;on)
          LineStyle(1;poly;2;blue)
          AxisOptions(y;;on) // y-Achse rechts
          GridLocation(xy;none) // kein Raster
     CloseChart()
CloseDrawing()
```



CloseChart()

`CloseChart()` besitzt keine Argumente und schließt die Befehlsfolge innerhalb eines Diagramms ab. Falls nur ein einziges Diagramm definiert wird, kann auf `CloseChart()` verzichtet werden.

Daten

Zur Eingabe und Verwaltung der Diagrammdaten stehen sechs Funktionen zur Verfügung. Sie dienen:

- zur Eingabe der Diagrammdaten
- zum Import und Export der Diagrammdaten
- zur Behandlung ungültiger oder nicht vorhandener Werte

ChartData(serie1;serie2...)

Die Funktion `ChartData()` dient zur Übergabe der Diagrammwerte an xmCHART. Die Zahlen werden üblicherweise mittels FileMaker Pro Scripts aus den Datenbankfeldern in die Funktion `ChartData()` kopiert. In der Beispielsammlung *xmBeispiele* wird darauf detailliert eingegangen.
Jede Serie besteht aus einer Folge von Zahlen, welche durch Leerzeichen, Tabs oder Zeilenumbrüche getrennt sind. Dezimalzahlen können sowohl durch Dezimalpunkt als auch Dezimalkomma dargestellt werden. Beispiel:

```
ChartData(-10 2.31 0 -,69 0.01 -8,00 .0) //erlaubt
```

E-Format und Tausender-Trennzeichen sind nicht erlaubt. Beispiel:

```
ChartData(-1.23E08 12.345,67) // nicht erlaubt!
```

Werden mehrere Werteserien an `ChartData()` übergeben, so sind diese durch einen Strichpunkt ";" zu trennen. Die Bedeutung der einzelnen Werteserien in `ChartData()` ist abhängig vom darzustellenden Diagramm. Bei den eindimensionalen Diagrammfunktionen `AreaChart()`, `BarChart()`, `BoxPlot()`, `GanttChart()`, `Histogram()`, `LineChart()`, `RadarChart()` und `ScatterChart()` entspricht eine Werteserie in `ChartData()` genau einer Diagrammserie. Bei den zweidimensionalen Diagrammfunktionen `LineChart2D()`, `ScatterChart2D()` und `PolarChart()` enthält die erste Werteserie in `ChartData()` die x-Werte, die zweite Werteserie die y-Werte der 1. Diagrammserie, die 3. und 4. Werteserie die x- und y-Werte der 2. Diagrammserie usw.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13  6 2; // 1.Serie
            18 12 19 24 15 15 6) // 2.Serie
  LineChart(symbol)
  SymbolStyle(all;bullet;4)
CloseDrawing()

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13  6 2; // x-Werte
            18 12 19 24 15 15 6) // y-Werte
  LineChart2D(symbol)
  SymbolStyle(all;bullet;4)
CloseDrawing()
```

Spezielle Zuordnungen zwischen Werteserien und Diagrammserien, wie zum Beispiel bei Hoch/Tief- oder Blasendiagrammen, werden im Abschnitt *Diagramme* erläutert.

ChartDataOptions(koordReihenfolge)

Standardmäßig werden die Werte einer Diagrammserie hintereinander eingegeben (*koordReihenfolge=xxyy*). Beispiel:

```
ChartData(2 3 -10  5  2; // Serie 1
          3 9  12 -3 11) // Serie 2
```

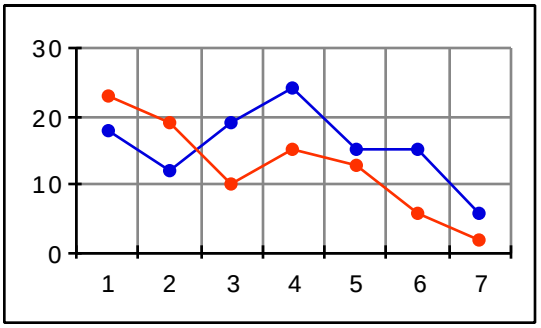
Bisweilen erweist es sich jedoch vorteilhaft, die Daten in transponierter Form einzugeben, d.h. die Zeilen und Spalten sind vertauscht angeordnet (*koordReihenfolge=xyxy*). Beispiel:

```
ChartData(2  3;
          3  9;
        -10 12;
          5 -3;
          2 11)
```

So können, zum Beispiel in einem FileMaker Pro Script, durch eine einzige Schleife gleichzeitig mehrere Werteserien aus einer Folge von Datensätzen in transponierter Form an `ChartData()` übergeben werden. Andernfalls, nicht transponiert, wird für jede Werteserie eine eigene Schleife über die erforderlichen FileMaker Pro Datensätze benötigt. Zu beachten ist, dass die Funktion `ChartDataOptions()` *vor* der Funktion `ChartData()` anzuführen ist.

Beispiele:

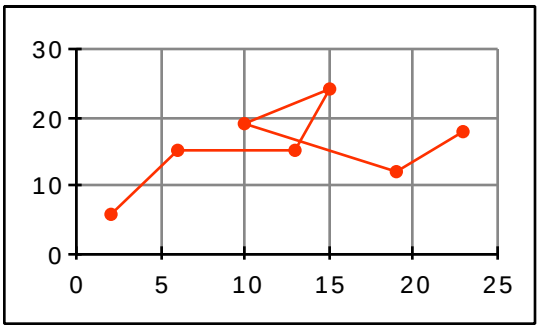
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartDataOptions(xyxy) // vor ChartData()!
     ChartData(23 18;
               19 12;
               10 19;
               15 24;
               13 15;
                6 15;
                2  6)
     LineChart(symbol;on)
     SymbolStyle(1;bullet;4)
     SymbolStyle(2;bullet;4)
CloseDrawing()
```



Die folgenden zwei Scripts führen zum gleichen Ergebnis:

```
// Eingabedaten nicht transponiert (default)
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 19 10 15 13  6 2; // x-Werte
               18 12 19 24 15 15 6) // y-Werte
     LineChart2D(symbol)
     SymbolStyle(1;bullet;4)
CloseDrawing()
```

```
// Eingabedaten transponiert
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartDataOptions(xyxy) // vor ChartData()!
     ChartData(23 18;
               19 12;
               10 19;
               15 24;
               13 15;
                6 15;
                2  6)
     LineChart2D(symbol)
     SymbolStyle(1;bullet;4)
CloseDrawing()
```

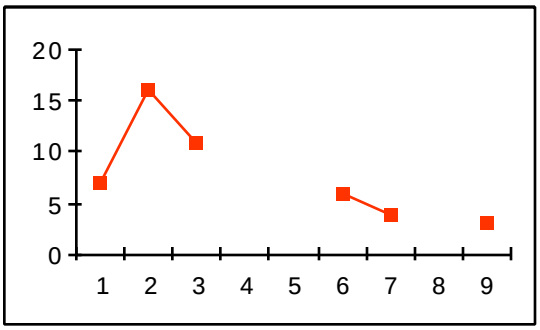


ChartDataUpperLimits(maxWert1;maxWert2...)
ChartDataLowerLimits(minWert1;minWert2...)

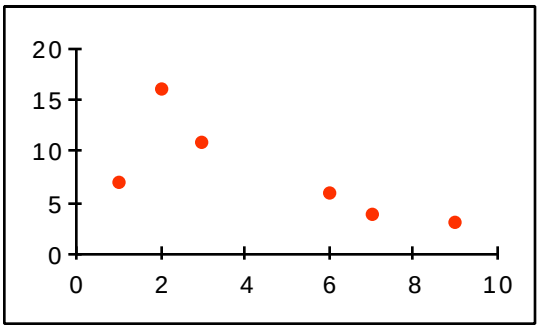
Die beiden Funktionen ChartDataUpperLimits() und ChartData-LowerLimits() ermöglichen zu jeder Werteserie einen oberen und unteren Grenzwert zu definieren, d.h. alle Werte, welche kleiner als *minWert* bzw. größer als *maxWert* sind, werden nicht dargestellt. Eine typische Anwendungsmöglichkeit ist die Darstellung unvollständiger Werteserien. Fehlende Daten, wie z.B. fehlende Meßwerte, werden durch "ungültige" Werte ergänzt, d.h. durch Werte, welche kleiner als *minWert* oder größer als *maxWert* sind. Zu beachten ist, dass die Funktionen ChartDataUpperLimits() und ChartDataLowerLimits() *nach* der Funktion ChartData() anzuführen sind.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     // -1..."fehlende Werte"
     ChartData(7 16 11 -1 -1 6 4 -1 3)
     ChartDataLowerLimits(0.0) // nach ChartData()!
     LineChart(symbol;on)
     SymbolStyle(1;square;4;;red)
     GridLocation(;none)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     // 9999..."fehlende Werte"
     ChartData(1  2  3    4   5  6 7    8 9;
               7 16 11 9999 9999 6 4 9999 3)
     ChartDataUpperLimits(1000) // nach ChartData()!
     ScatterChart2D()
     SymbolStyle(1;bullet;4)
     GridLocation(;none)
CloseDrawing()
```



ChartDataRead(dateiname;istTransponiert; serientrennzeichen;elementtrennzeichen)

Anstelle die Diagrammwerte via `ChartData()` direkt aus FileMaker Pro zu übernehmen, können diese auch aus einer Textdatei importiert werden. Dadurch besteht die Möglichkeit, auch sehr große Datenmengen zu importieren, da die 64kB Beschränkung von FileMaker Pro Feldern umgangen werden kann.
Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad festgelegt; Details dazu finden sich im Abschnitt *Ausgabe*. Standardmäßig werden die Daten zeilenweise eingelesen. Falls das 2. Argument *istTransponiert=on* gesetzt wird, werden die Daten spaltenweise eingelesen, d.h. eine Datenserie entspricht einer Spalte in der Datei. Durch die Argumente *serientrennzeichen* und *elementtrennzeichen* können benutzerdefinierte Trennzeichen zwischen den Serien und zwischen den einzelnen Werten festgelegt werden. Wenn nicht anders vorgegeben, sind Serien durch einen Zeilenumbruch "\n" zu trennen und die einzelne Werte durch ein Tabulatorzeichen "\t".
Dezimalzahlen können sowohl mit Dezimalpunkt als auch mit Dezimalkomma angeführt werden; Tausender-Trennzeichen und E-Format sind jedoch nicht erlaubt. Die Funktion `ChartDataOptions()` wird in Verbindung mit `ChartDataRead()` ignoriert. Beispiele:

```
ChartDataRead("Macintosh HD:Daten:Plotdaten.dat")
ChartDataRead("C:\\Programme\\Daten\\plotdata.txt";;";";" ")
```

Die Funktion `ChartDataRead()` kann beliebig platziert werden, jedoch immer innerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()`. `ChartDataRead()` kann auch mehrfach aufgerufen werden.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     LineChart(symbol;on)
     SymbolStyle(1;bullet;4)
     SymbolStyle(2;bullet;4)
     ChartDataRead("Daten")
CloseDrawing()
```

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(5;5;120;115;off)
          ChartDataRead("Daten-1.txt")
          BarChart()
     CloseChart()
     OpenChart(120;0;75;75;off)
          ChartDataRead("Daten-2.txt")
          PieChart(label)
     CloseChart()
     OpenChart(120;50;75;75;off)
          ChartDataRead("Daten-3.txt")
          PieChart(label)
     CloseChart()
CloseDrawing()
```

ChartDataWrite(dateiname;dateiflag;dateityp; istTransponiert;serientrennzeichen; elementtrennzeichen;format1;format2...)

Mit der Funktion `ChartDataWrite()` ist es möglich, die Diagrammwerte in eine Textdatei zu sichern. Die ersten drei Argumente *dateiname*, *dateiflag* und *dateityp* werden detailliert im Abschnitt *Ausgabe* erklärt. Standardmäßig werden die Daten zeilenweise weggeschrieben, d.h. einer Werteserie entspricht eine Ausgabezeile. Falls das 4. Argument *istTransponiert=on* gesetzt wird, werden die Daten spaltenweise ausgegeben, d.h. einer Werteserie entspricht eine Ausgabespalte. Weiters können durch die Argumente *serientrennzeichen* und *elementtrennzeichen* benutzerdefinierte Trennzeichen festgelegt werden. Wenn nicht anders vorgegeben, werden Serien durch einen Zeilenumbruch "\n" getrennt und die einzelne Werte durch ein Tabulatorzeichen "\t". Optional können den Werteserien individuelle Formatanweisungen zugeordnet werden. Die Formatanweisungen werden periodisch wiederholt, falls weniger Formatanweisungen als Werteserien vorhanden sind. Wenn keine Formatanweisungen vorgegeben werden, wird das Defaultformat "|u|" verwendet. Sämtliche Formatanweisungen inkl. zahlreicher Beispiele finden sich in *xmReferenz*.
Alle Zahlen werden ohne Tausender-Trennzeichen ausgegeben, alle Dezimalzahlen stets mit einem Dezimalpunkt "." und nicht mit einem Dezimalkomma ",".

Beispiele:

```
ChartDataWrite("Macintosh HD:Plots:ExportDaten";replace)
ChartDataWrite("Daten.txt";;"ttxt";;";";" ";"|i0|";"|f2|")
ChartDataWrite("C:\\Programme\\Daten\\export.txt";;;on)
```

Die Funktion `ChartDataWrite()` kann sowohl vor als auch nach der Funktion `ChartData()` angeführt werden, jedoch immer innerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()`.
`ChartDataWrite()` kann auch mehrfach aufgerufen werden.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartDataWrite("Daten";replace;;on;;;"|f1|";"|f2|")
     ChartData(23 19 10 15 13  6 2; // x-Werte
               18 12 19 24 15 15 6) // y-Werte
     LineChart2D(symbol)
     SymbolStyle(1;bullet;4)
CloseDrawing()
```

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(5;5;120;115;off)
          ChartData(3 2 1 5 3)
          ChartDataWrite("Daten-1.txt";replace)
          BarChart()
     CloseChart()
     OpenChart(120;0;75;75;off)
          ChartData(13 9 13)
          ChartDataWrite("Daten-2.txt";replace)
          PieChart(label)
     CloseChart()
     OpenChart(120;50;75;75;off)
          ChartData(1.5 2 1,5)
          ChartDataWrite("Daten-3.txt";replace)
          PieChart(label)
     CloseChart()
CloseDrawing()
```

Diagramme

xmCHART stellt eine Vielzahl von Diagrammarten zur Verfügung, welche grob in folgende Kategorien eingeteilt werden können:

- Punktediagramme (1- und 2-dimensional)
- Liniendiagramme (1- und 2-dimensional)
- Flächendiagramme
- Balkendiagramme
- Blasendiagramme (1- und 2-dimensional)
- Kuchendiagramme
- Polar- und Radardiagramme
- Börsendiagramme
- Statistische Diagramme

Alle Diagramme können durch zahlreiche Optionen gestaltet und variiert werden. Nachfolgend werden sämtliche Diagrammarten ausführlich erläutert und durch eine Vielzahl anschaulicher Beispiele ergänzt.

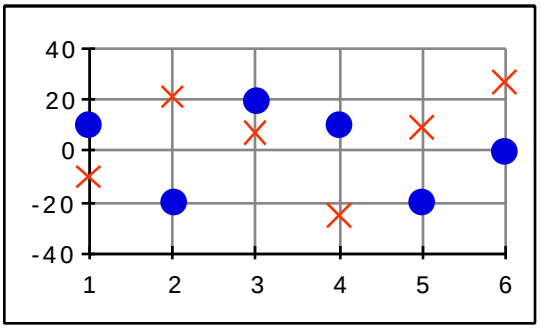
Punktediagramme:

Zur Darstellung von Punktediagrammen stehen zwei Funktionen zur Verfügung:

ScatterChart(darstellung;intervalleVerschieben)

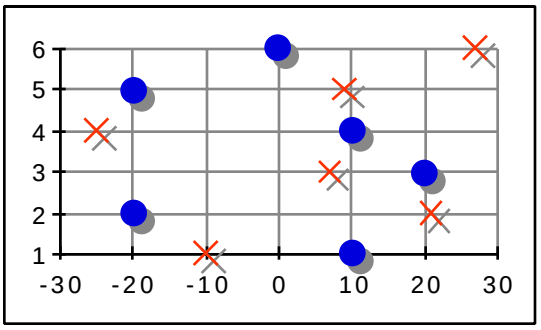
Die Funktion `ScatterChart()` dient zur Darstellung von 1-dimensionalen Punktediagrammen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(-10  21  7 -25   9 27; // 1.Serie
               10 -20 20  10 -20  0) // 2.Serie
     ScatterChart()
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Symbole mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

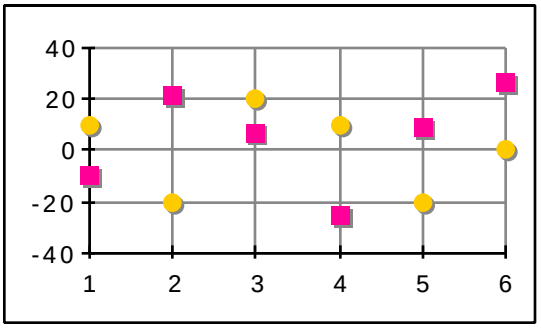
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(-10  21  7 -25   9 27; // 1.Serie
               10 -20 20  10 -20  0) // 2.Serie
     ScatterChart(shadow+horizontal)
CloseDrawing()
```



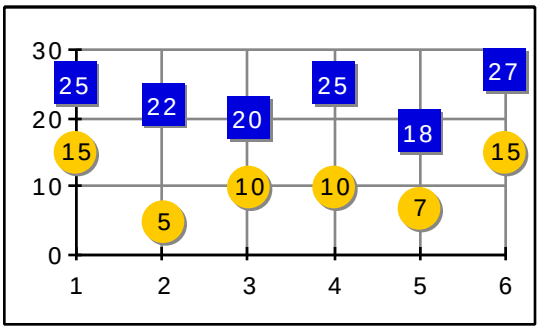
Die Darstellung der Symbole und Schatten kann durch die Stilfunktionen `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(-10  21  7 -25   9 27; // 1.Serie
               10 -20 20  10 -20  0) // 2.Serie
     ScatterChart(shadow)
     SymbolStyle(1;square;6;;purple)
     SymbolStyle(2;bullet;6;;darkYellow)
     ShadowStyle(all;1;gray)
CloseDrawing()
```

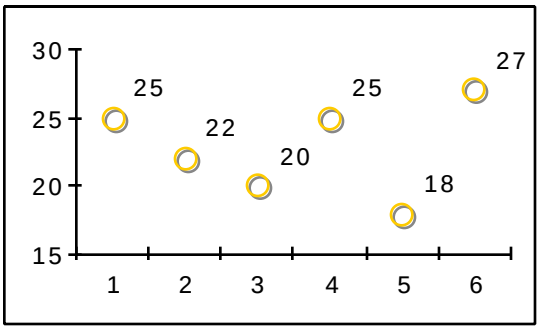



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25 22 20 25 18 27; // 1.Serie
               15  5 10 10  7 15) // 2.Serie
     ScatterChart(shadow+label)
     SymbolStyle(1;square;15;;blue)
     SymbolStyle(2;bullet;15;;darkYellow)
     ShadowStyle(all;1;gray)
     LabelOptions(all;centerCenter)
     LabelStyle(1;;;bold;white)
CloseDrawing()
```



Bei eindimensionalen Diagrammen erweist es sich bisweilen als vorteilhaft, die Symbole um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies geschieht durch Aktivieren des 2. Arguments *intervalleVerschieben=on*. Beispiel:

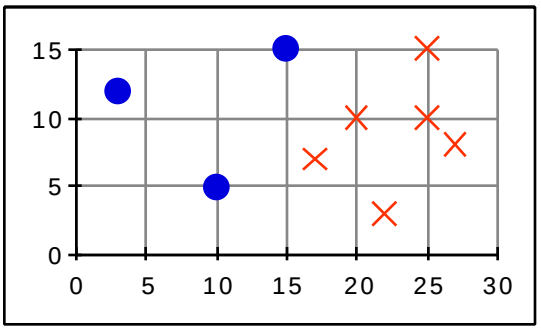
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25 22 20 25 18 27)
     ScatterChart(shadow+label;on)
     SymbolStyle(1;circle;8;;darkYellow)
     ShadowStyle(all;1;gray)
     GridLocation(all;none)
CloseDrawing()
```



ScatterChart2D(darstellung)

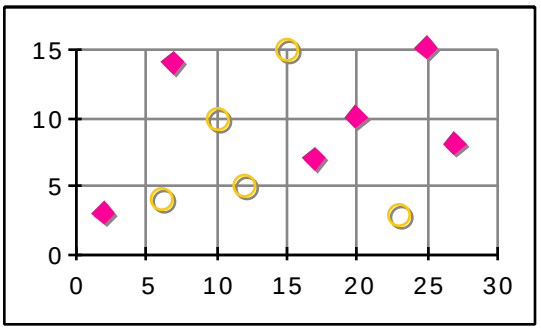
Die Funktion `ScatterChart2D()` ermöglicht die Darstellung von zweidimensionalen Punktediagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte zur ersten Symbolserie, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte zur zweiten Symbolserie usw. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25 22 20 25 17 27; // x-Werte 1.Serie
               15  3 10 10  7  8; // y-Werte 1.Serie
                3 10 15;          // x-Werte 2.Serie
               12  5 15)          // y-Werte 2.Serie
     ScatterChart2D()
CloseDrawing()
```



Wie bei den eindimensionalen Punktediagrammen können auch hier durch das Argument *darstellung* die Symbole mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25  2 20  7 17 27; // x-Werte 1.Serie
               15  3 10 14  7  8; // y-Werte 1.Serie
               23 10 15 12  6;    // x-Werte 2.Serie
                3 10 15  5  4)    // y-Werte 2.Serie
     ScatterChart2D(shadow)
     SymbolStyle(1;diamond;7;;purple)
     SymbolStyle(2;circle;8;;darkYellow)
     ShadowStyle(all;1;gray)
CloseDrawing()
```



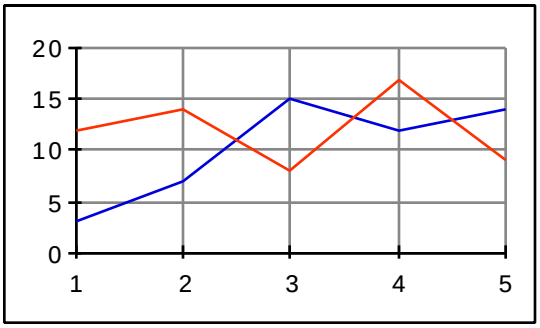
Liniendiagramme:

Zur Darstellung von Liniendiagrammen stehen zwei Funktionen zur Verfügung:

LineChart(darstellung;intervalleVerschieben)

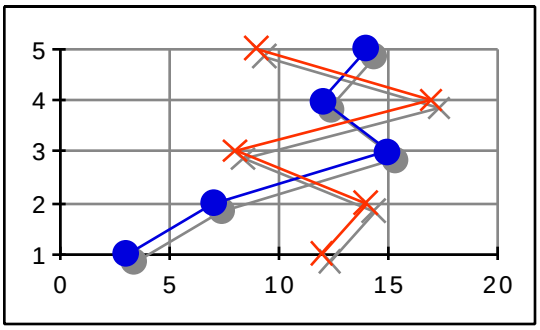
Die Funktion `LineChart()` dient zur Darstellung von eindimensionalen Liniendiagrammen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 14  8 17  9;
               3  7 15 12 14)
     LineChart()
CloseDrawing()
```



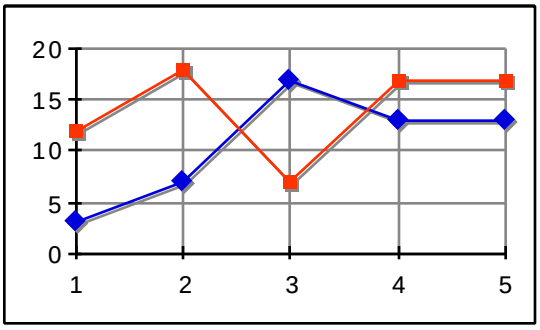
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Linien mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 14 8 17 9; 3 7 15 12 14)
     LineChart(shadow+horizontal+symbol)
CloseDrawing()
```

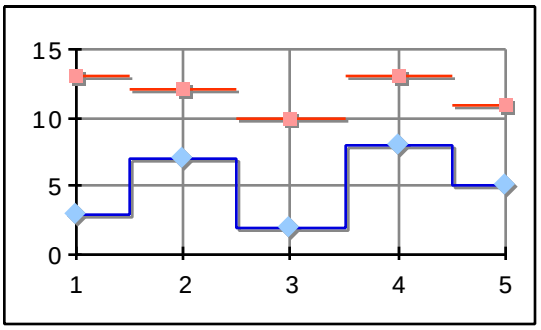


Die Darstellung der Linien, Symbole und Schatten kann durch die Stilfunktionen `LineStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

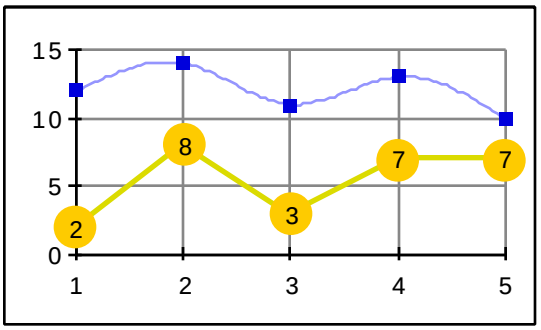
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 18 7 17 17; 3 7 17 13 13)
     LineChart(shadow+symbol)
     SymbolStyle(1;square;4;;red)
     SymbolStyle(2;diamond;6;;blue)
     ShadowStyle(all;1;gray)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(13 12 10 13 11; 3  7 2 8 5)
     LineChart(shadow+symbol)
     LineStyle(1;jump)
     LineStyle(2;step)
     SymbolStyle(1;square;4;;lightRed)
     SymbolStyle(2;diamond;6;;lightBlue)
     ShadowStyle(all;1;gray)
CloseDrawing()
```

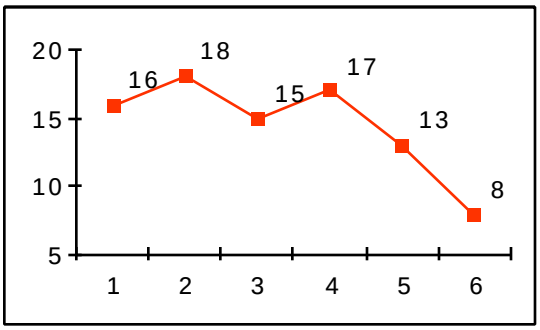



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 14 11 13 10; 2 8 3 7 7)
     LineChart(symbol+label)
     LineStyle(1;smooth;1;150 150 255)
     LineStyle(2;poly;2;220 220 0)
     SymbolStyle(1;square;4;;blue)
     SymbolStyle(2;bullet;15;;darkYellow)
     LabelTexts(1;" ") // keine Beschriftung
     LabelOptions(2;centerCenter)
CloseDrawing()
```



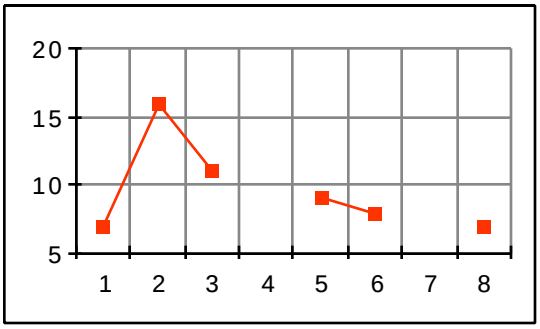
Bei eindimensionalen Liniendiagrammen besteht die Möglichkeit, die Polygonpunkte um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol+label;on)
     SymbolStyle(1;square;4;;red)
     GridLocation(all;none)
CloseDrawing()
```



Zusätzlich besteht die Möglichkeit, unter Verwendung der Funktionen `ChartDataLowerLimits()` und/oder `ChartDataUpperLimits()` unterbrochene Linienzüge darzustellen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 16 11 -1 9 8 -1 7)
     // Werte kleiner 0 werden nicht dargestellt
     ChartDataLowerLimits(0)
     LineChart(symbol;on)
     SymbolStyle(1;square;4;;red)
CloseDrawing()
```



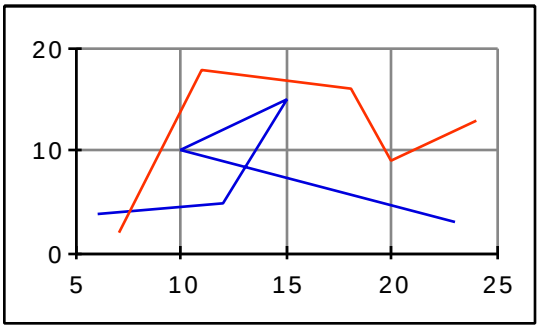
Die beiden Funktionen `ChartDataLowerLimits()` und `ChartDataUpperLimits()` werden im Abschnitt *Daten* erläutert.

LineChart2D(darstellung)

Die Funktion `LineChart2D()` ermöglicht die Darstellung von zweidimensionalen Liniendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte des ersten Linienzugs, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte des zweiten Linienzugs usw.

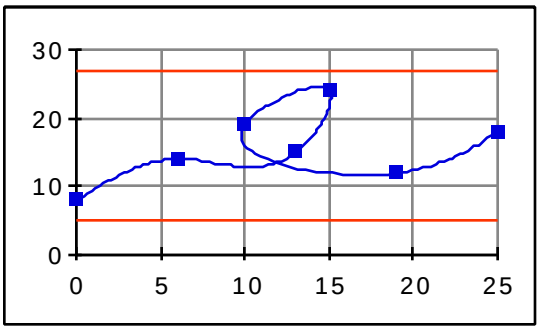
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 18 20 24; // x-Werte 1.Serie
                2 18 16  9 13; // y-Werte 1.Serie
               23 10 15 12  6; // x-Werte 2.Serie
                3 10 15  5  4) // y-Werte 2.Serie
     LineChart2D()
CloseDrawing()
```



Wie bei den eindimensionalen Liniendiagrammen können auch hier durch das Argument *darstellung* die Linien mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25 19 10 15 13  6 0; // x-Werte 1.Serie
               18 12 19 24 15 14 8; // y-Werte 1.Serie
                0 25; 27 27; // obere Begrenzungslinie
                0 25;  5  5) // untere Begrenzungslinie
     LineChart2D(symbol)
     LineStyle(all;poly;1;red)
     LineStyle(1;smooth;1;blue)
     SymbolStyle(all;none)
     SymbolStyle(1;square;4;;blue)
CloseDrawing()
```

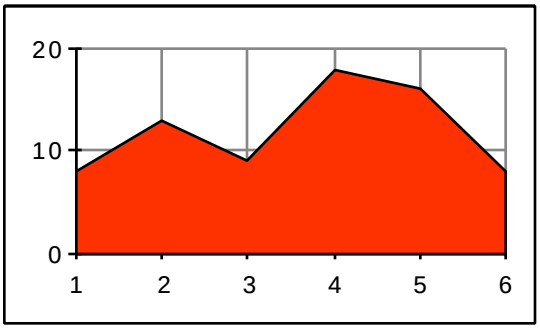


Flächendiagramme:

AreaChart(darstellung;intervalleVerschieben)

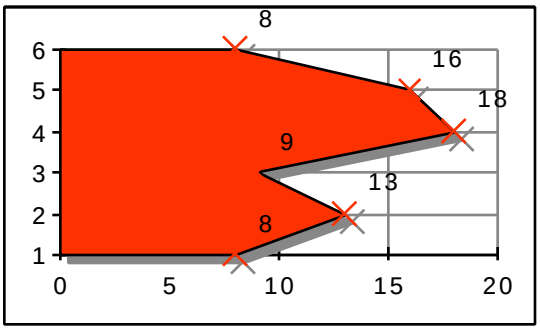
Die Funktion `AreaChart()` dient zur Darstellung von eindimensionalen Flächendiagrammen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(8 13 9 18 16 8)
     AreaChart()
CloseDrawing()
```



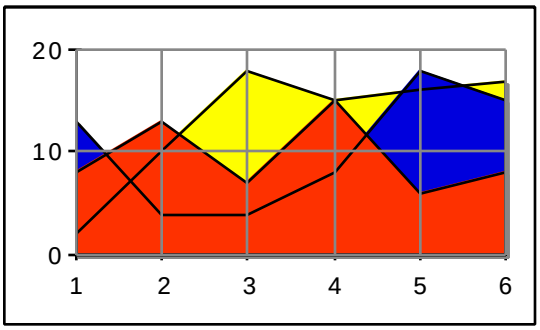
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(8 13 9 18 16 8)
     AreaChart(symbol+label+horizontal+shadow)
CloseDrawing()
```

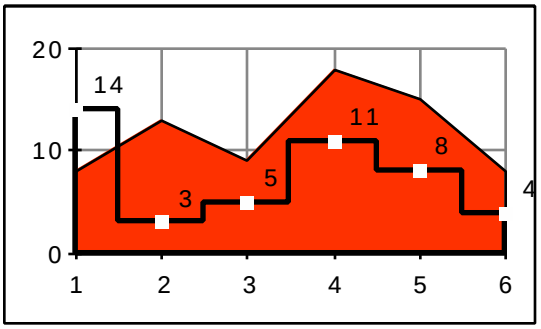


Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

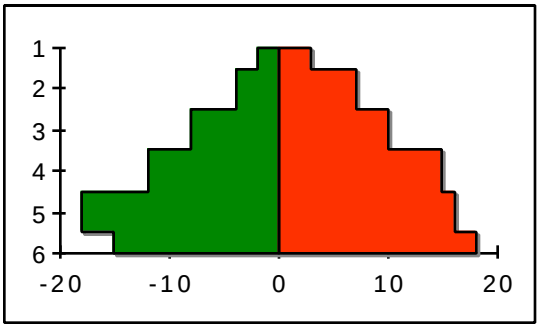
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8 13  7 15  6  8;
               13  4  4  8 18 15;
                2 10 18 15 16 17)
     AreaChart(shadow)
     ShadowStyle(all;1;gray)
     GridLocation(all;front)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(8 13 9 18 15 8; 14 3 5 11 8 4)
     AreaChart(symbol+label)
     SymbolStyle(1;none)
     LabelTexts(1;" ") // keine Beschriftung
     BorderStyle(2;step;2)
     FillStyle(2;;transparent)
     SymbolStyle(2;square;4;;white)
CloseDrawing()
```

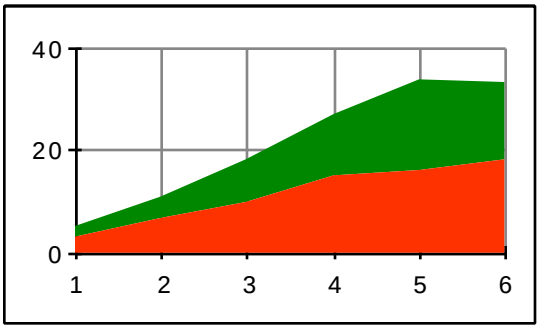



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 3   7  10  15  16  18;
               -2  -4  -8 -12 -18 -15)
     AreaChart(shadow+horizontal)
     BorderStyle(all;step)
     FillStyle(2;green)
     ShadowStyle(all;1;gray)
     GridLocation(all;none)
     ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()
```

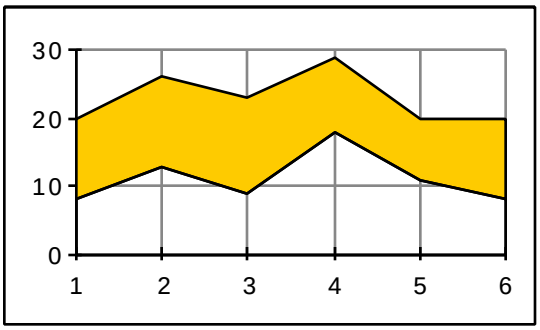


Weiters können eindimensionale Flächendiagramme gestapelt (*darstellung=stacked*) und proportional (*darstellung=proportional*) dargestellt werden. Proportionale Diagramme werden automatisch gestapelt. Die Beschriftung gestapelter und proportionaler Diagramme wird ausführlich anhand zahlreicher Beispiele im Abschnitt *Stile* erläutert. Beispiele:

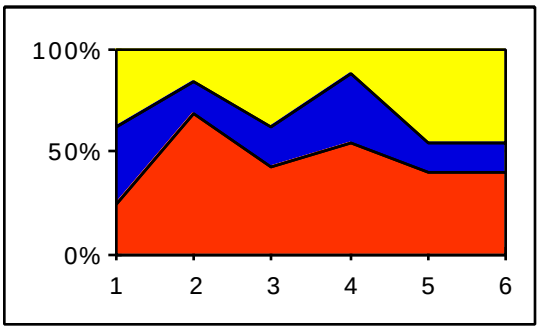
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 7 10 15 16 18;
               2 4  8 12 18 15)
     AreaChart(stacked)
     BorderStyle(all;none)
     FillStyle(2;green)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8 13  9 18 11  8;
               12 13 14 11  9 12)
     AreaChart(stacked)
     FillStyle(1;;transparent)
     FillStyle(2;darkYellow)
CloseDrawing()
```

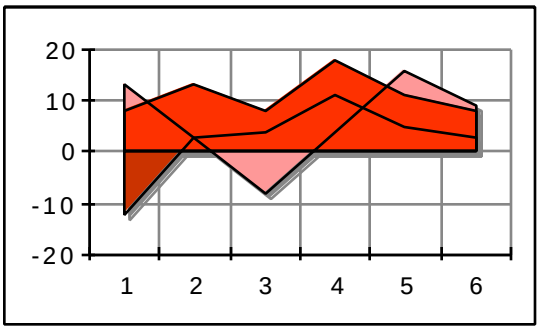


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8 13 9 18 14 8;
               12  3 4 11  5 3;
               12  3 8  4 16 9)
     AreaChart(stacked+proportional)
     Scaling(y;percent)
CloseDrawing()
```



Bei eindimensionalen Diagrammen besteht optional die Möglichkeit, die Polygonpunkte um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen, sondern in der Mitte der Intervalle liegen. Dies geschieht durch Aktivieren des 2. Arguments *intervalleVerschieben=on*. Beispiel:

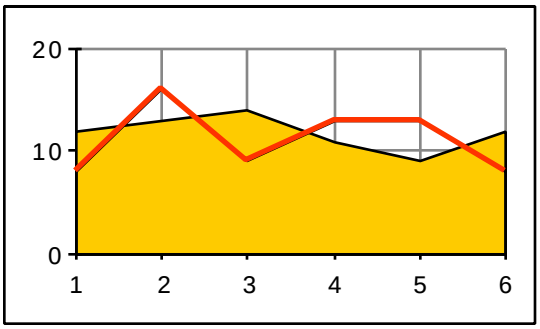
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8 13  8 18 11 8;
              -12  3  4 11  5 3;
               13  3 -8  4 16 9)
     AreaChart(shadow;on)
     FillStyle(2;darkRed)
     FillStyle(3;lightRed)
     ShadowStyle(all;2;gray)
CloseDrawing()
```



AreaChartOptions(linienstilVerwenden)

Optional kann durch Aktivieren des Arguments *linienstilVerwenden=on* die Berandung durch einen Linienzug ergänzt werden. Im Gegensatz zur standardmäßigen Berandung, welche die gesamte Fläche umschließt, d.h. inklusive der Seitenränder und der Grundlinie, bezieht sich der Linienzug nur auf den Bereich zwischen den Diagrammwerten. Das Aussehen des Polygonzugs kann durch die Funktion `LineStyle()` gestaltet werden. Dabei wird in der Funktion `LineStyle()` das zweite Argument *darstellung*, welches üblicherweise zur Kontrolle des Kurvenverlaufs dient, in diesem Zusammenhang ignoriert. Eine mögliche Anwendung ist die gleichzeitige Darstellung eines Flächen- und Liniendiagramms; dazu wird die Füllung einer Serie *transparent* gesetzt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8 16  9 13 13  8; // Linie
               12 13 14 11  9 12) // Fläche
     AreaChart()
     AreaChartOptions(on) // nach AreaChart()!
     LineStyle(1;;2;red)
     LineStyle(2;;0)
     FillStyle(1;;transparent) // keine Füllung
     FillStyle(2;darkYellow)
CloseDrawing()
```



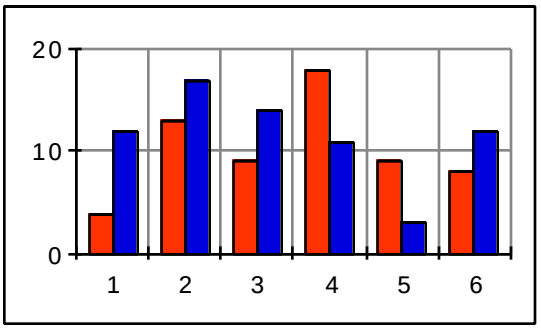
Zu beachten ist, dass die Funktion `AreaChartOptions()` nach der Funktion `AreaChart()` anzuführen ist. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen.*

Balkendiagramme:

BarChart(darstellung;kategorienabstand;serienabstand; balkentiefe)

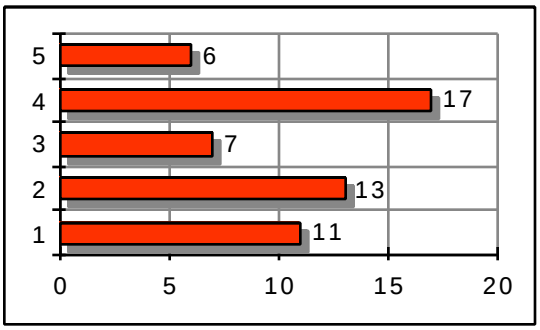
Die Funktion `BarChart()` dient zur Darstellung von Balkendiagrammen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18 9  8; // "rote" Serie
               12 17 14 11 3 12) // "blaue" Serie
     BarChart()
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Balken mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

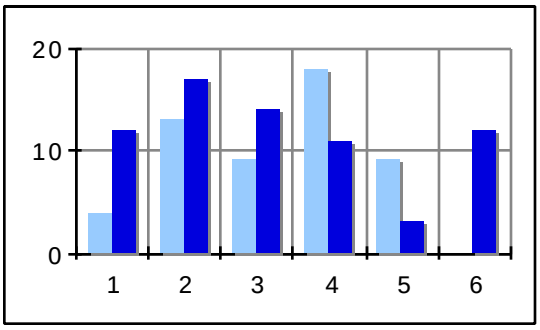
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(11 13 7 17 6)
     BarChart(label+shadow+horizontal)
CloseDrawing()
```



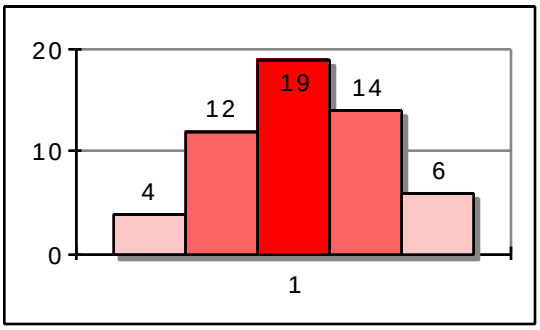
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18 9;
                12 17 14 11 3 12)
     BarChart(shadow)
     FillStyle(1;lightBlue)
     BorderStyle(all;none)
     ShadowStyle(all;1;gray)
CloseDrawing()
```

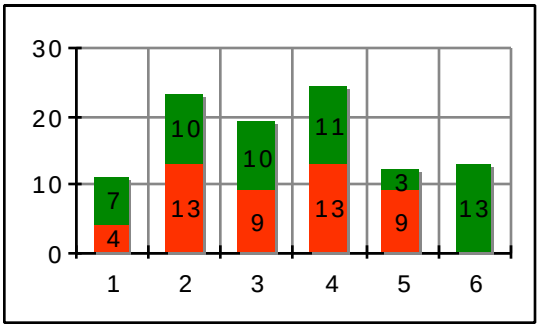


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4; 12; 19; 14; 6)
     BarChart(shadow+label)
     FillStyle(1;255 200 200)
     FillStyle(2;255 100 100)
     FillStyle(3;255   0   0)
     FillStyle(4;255 100 100)
     FillStyle(5;255 200 200)
     ShadowStyle(all;2;gray)
CloseDrawing()
```

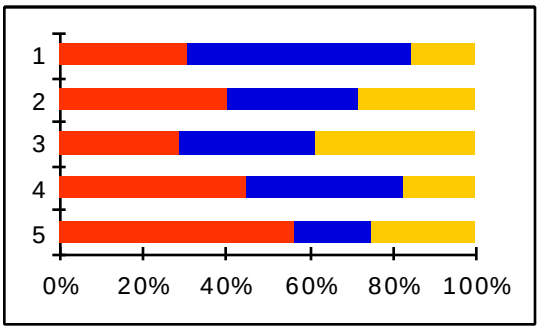


Weiters können Balkendiagramme gestapelt (*darstellung=stacked*) und proportional (*darstellung=proportional*) dargestellt werden. Proportionale Diagramme werden automatisch gestapelt. Die Beschriftung gestapelter und proportionaler Diagramme wird ausführlich anhand zahlreicher Beispiele im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(6 13  9 13 9;
               7 10 10 11 5 13)
     BarChart(shadow+stacked+label)
     FillStyle(2;green)
     BorderStyle(all;none)
     ShadowStyle(all;1;gray)
CloseDrawing()
```



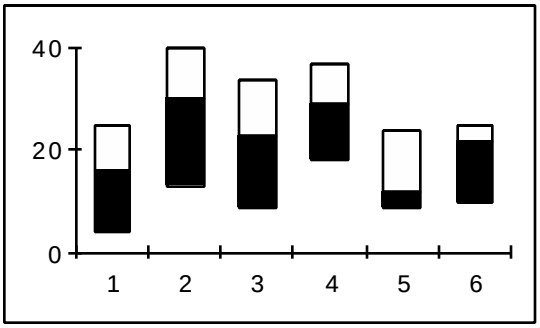
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13  9 13 9;
               7 10 10 11 3;
               2  9 12  5 4)
     BarChart(horizontal+proportional)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
     Scaling(x;percent)
     ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()
```



Zusätzlich besteht die Möglichkeit, durch Transparent-Setzen der ersten Datenserie, gleitende Balkendiagramme zu erstellen.

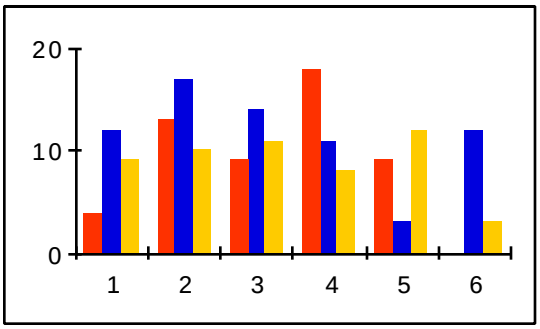
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9 10;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(stacked)
     BorderStyle(1;none)
     FillStyle(1;;transparent)
     FillStyle(2;black)
     FillStyle(3;white)
     GridLocation(all;none)
CloseDrawing()
```



Durch die beiden Argumente *kategorienabstand* und *serienabstand* können die Abstände zwischen den Balken kontrolliert werden. Beide Argumente sind in Prozent der Balkenbreite einzugeben. Standardmäßig ist der Abstand zwischen zwei Balkengruppen (Kategorien) genau eine Balkenbreite, d.h. *kategorienabstand=100*. Beispiel:

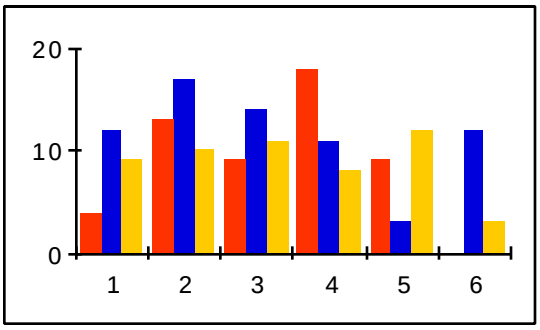
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;100) // Default-Abstand
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



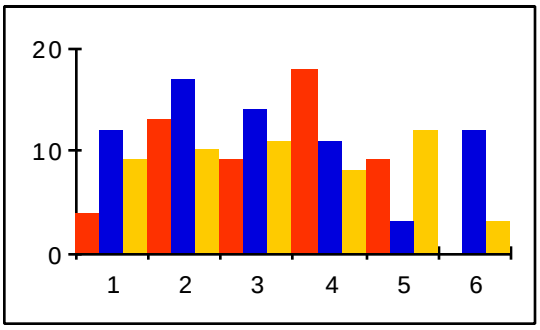
Der Abstand zwischen den Kategorien wird z.B. auf die Hälfte verkleinert durch *kategorienabstand=50*, oder komplett unterdrückt durch *kategorienabstand=0*.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;50)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```

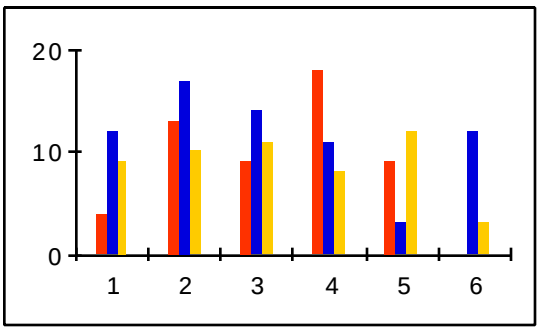



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;0)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



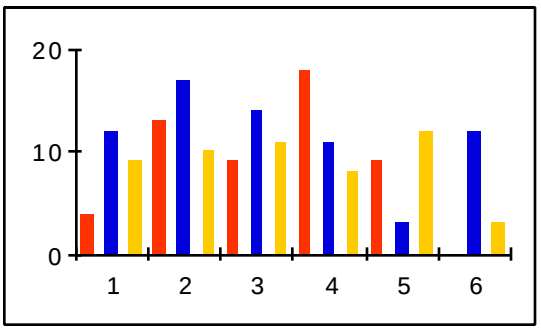
Durch z.B. *kategorienabstand=400* wird der Abstand zwischen zwei Balkengruppen auf das 4-fache der Balkenbreite vergrößert. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;400)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



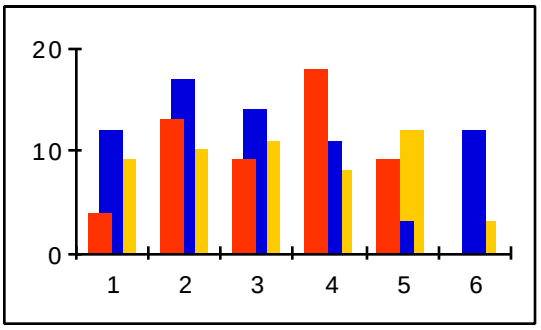
Bei nicht gestapelten Balken sind standardmäßig keine Lücken zwischen den Serien vorgesehen (*serienabstand=0*). Durch z.B. *serienabstand=100* wird zwischen den einzelnen Serien eine Lücke eingefügt. Die Größe der Lücke entspricht bei einem *serienabstand=100* der Balkenbreite. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;;100)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



Ein negativer *serienabstand* führt bei nicht gestapelten Balken zu einer teilweisen oder vollständigen Überlappung der Balken. Beispiel:

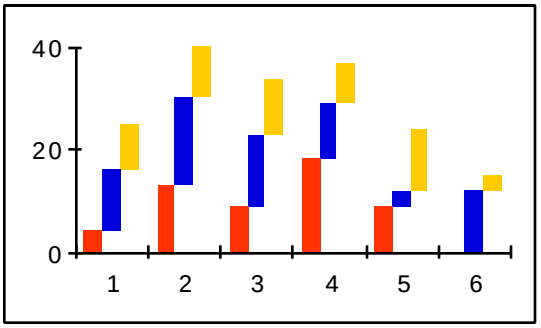
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(;;-50)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



Da gestapelte Balken nicht nebeneinander, sondern übereinander angeordnet sind, ist der *serienabstand* standardmäßig -100. Bei einem *serienabstand* ungleich -100, zum Beispiel *serienabstand=0*, werden die gestapelten Balken versetzt dargestellt.

Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18  9;
               12 17 14 11  3 12;
                9 10 11  8 12  3)
     BarChart(stacked;;0)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



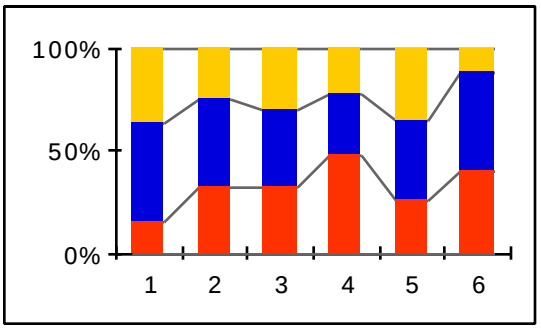
Das 4. Argument *balkentiefe* ist noch nicht implementiert.

BarChartOptions(verbindungslinien;referenzwert; farbteilung)

Durch Aktivieren des ersten Arguments *verbindungslinien=on* werden die — in diesem Fall üblicherweise gestapelten — Balken durch Linien verbunden. Die Gestaltung der Linien erfolgt durch die Funktion `LineStyle()`. Die Funktion `LineStyle()` wird ausführlich im Abschnitt *Stile* erläutert. Zu beachten ist, dass die Funktion `BarChartOptions()` nach der Funktion `BarChart()` anzuführen ist. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen.*

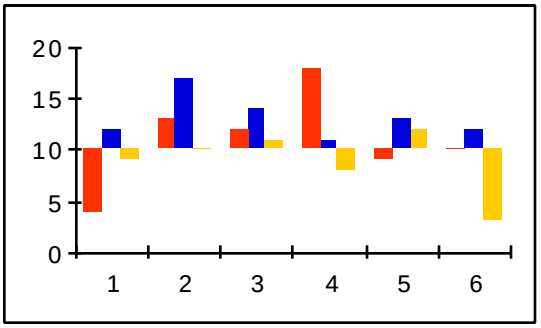
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13 12 18  9 10;
               12 17 14 11 13 12;
                9 10 11  8 12  3)
     BarChart(proportional)
     BarChartOptions(on)
     FillStyle(3;darkYellow)
     LineStyle(all;poly;1;darkGray)
     BorderStyle(all;none)
     GridLocation(all;none)
     Scaling(y;percent)
CloseDrawing()
```



Das zweite Argument *referenzwert* ermöglicht die Vorgabe einer benutzerdefinierten Grundlinie. Standardmäßig beginnen alle Balken bei Null, d.h. *referenzwert=0*. Beispiel:

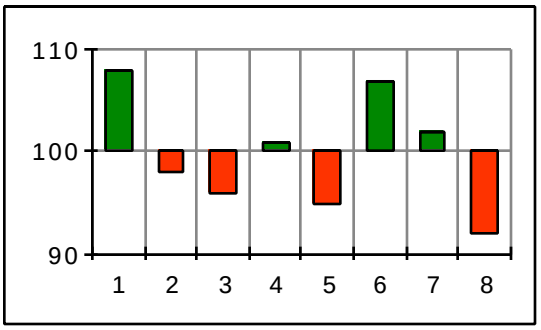
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13 12 18  9 10;
               12 17 14 11 13 12;
                9 10 11  8 12  3)
     BarChart()
     BarChartOptions(;10) // Referenzwert = 10
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```



Durch Aktivieren des dritten Arguments *farbteilung=on* werden die Werte, welche größer bzw. kleiner als der Referenzwert sind, durch Balken unterschiedlicher Farbe dargestellt. Das Aussehen der "positiven" Balken, d.h die Diagrammwerte sind größer als der Referenzwert, wird durch Stilfunktionen mit ungeraden Serienindizes festgelegt, das Aussehen der "negativen" Balken durch Stilfunktionen mit geradzahligen Serienindizes.

Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(108 98 96 101 95 107 102 92)
     BarChart()
     BarChartOptions(;100;on) // Referenzwert = 100
     FillStyle(1;green) // "positiv", ungerader Serienind.
     FillStyle(2;red)   // "negativ", gerader Serienindex
CloseDrawing()
```



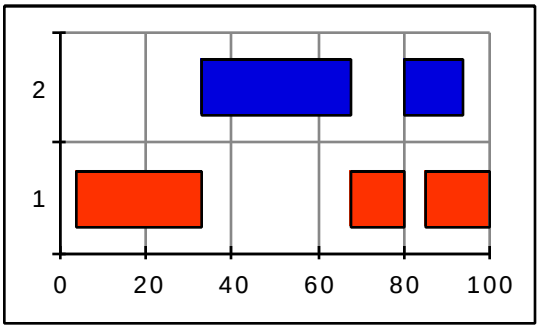
Zu beachten ist, dass die Farbteilung bei der Legende nicht unterstützt wird.

Gantt-Diagramme:

GanttChart(darstellung;kategorienabstand;balkentiefe)

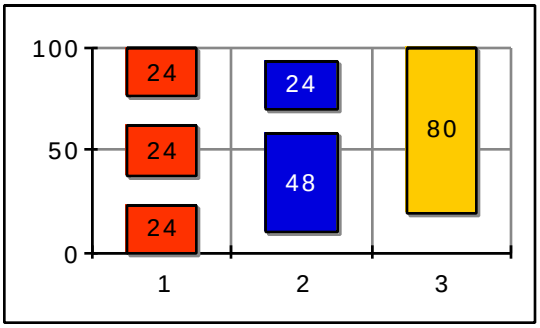
Die Funktion `GanttChart()` dient zur grafischen Aufbereitung von zeitabhängigen Aufgaben. Diese Art der Darstellung wurde von Henry L. Gantt 1917 erstmals verwendet. Jede Aufgabe wird durch die Eingabe von mindestens zwei Werten in `ChartData()` festgelegt; der erste Wert definiert den Beginn einer Aufgabe, der zweite Wert das Ende. Durch die Vorgabe von mehr als zwei Werten pro Serie kann eine Aufgabe in mehrere Zeitabschnitte unterteilt werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 33 68 80 85 100; // Proj 1 (3 Teilber.)
                33 68 80 94)       // Proj 2 (2 Teilber.)
     GanttChart()
CloseDrawing()
```



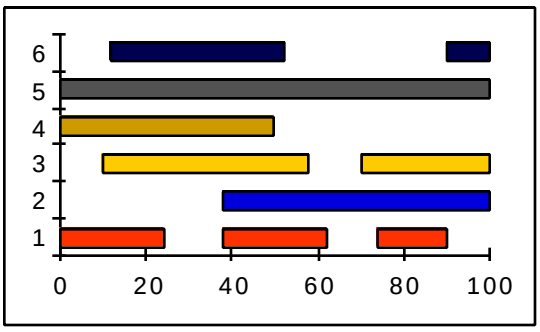
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Balken durch Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 0 24  38 62  76 100;
                10 58  70 94;
                20 100)
     GanttChart(label+shadow+horizontal)
     FillStyle(3;darkYellow)
     LabelStyle(2;;;bold;white)
     ShadowStyle(all;1;gray)
CloseDrawing()
```

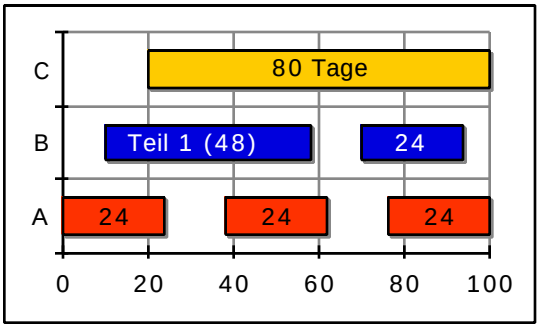


Die Darstellung der Füllungen, Berandungen und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 0  24  38 62  74 90;
               38 100;
               10  58  70 100;
                0  50;
                0 100;
               12  52  90 100)
     GanttChart()
     FillStyle(3;darkYellow)
     GridLocation(all;none)
CloseDrawing()
```

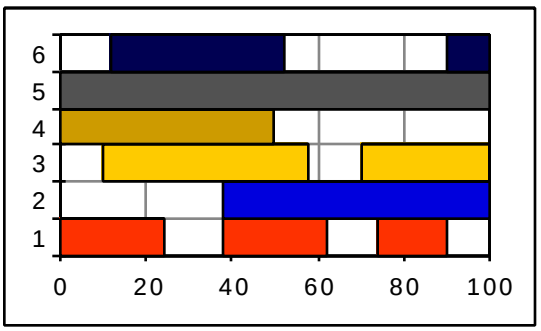


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 0  24  38 62  76 100;
                10  58  70 94;
                20 100)
     GanttChart(label+shadow)
     FillStyle(3;darkYellow)
     LabelStyle(2;;;bold;white)
     LabelTexts(2;"Teil 1 (|u|)";"|u|")
     LabelTexts(3;"|u| Tage")
     ShadowStyle(all;1;gray)
     AxisMajorTickLabelTexts(y;"A";"B";"C")
CloseDrawing()
```

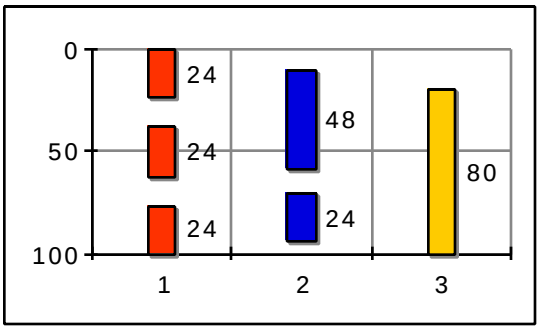


Durch das 2. Argument *kategorienabstand* kann der Abstand zwischen den Balken kontrolliert werden. Das Argument *kategorienabstand* ist in Prozent der Balkenbreite einzugeben. Standardmäßig ist der Abstand zwischen zwei Balken (Kategorien) genau eine Balkenbreite, d.h. *kategorienabstand=100*. Bei einem *kategorienabstand* kleiner 100 wird der Balkenabstand kleiner und die Balken breiter, bei einem *kategorienabstand* größer 100 wird der Abstand größer und die Balken schmäler. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 0  24  38 62  74 90;
                38 100;
                10  58  70 100;
                 0  50;
                 0 100;
                12  52  90 100)
     GanttChart(;0)
     FillStyle(3;darkYellow)
     GridFrame()
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 0  24  38 62  76 100;
                10  58  70 94;
                20 100)
     GanttChart(label+shadow+horizontal;400)
     FillStyle(3;darkYellow)
     ShadowStyle(all;1;gray)
     ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()
```



Das 3. Argument *balkentiefe* ist zur Zeit noch nicht implementiert.

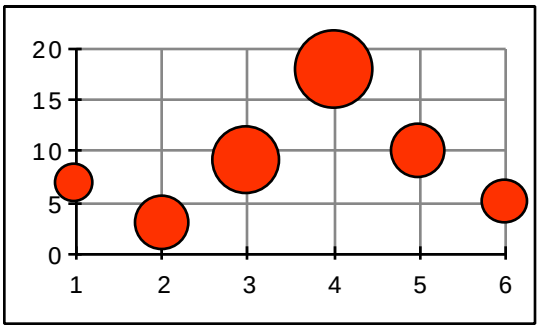
Blasendiagramme:

Blasendiagramme (bubble charts) können sowohl ein- als auch zweidimensional dargestellt werden.

BubbleChart(darstellung;intervalleVerschieben)

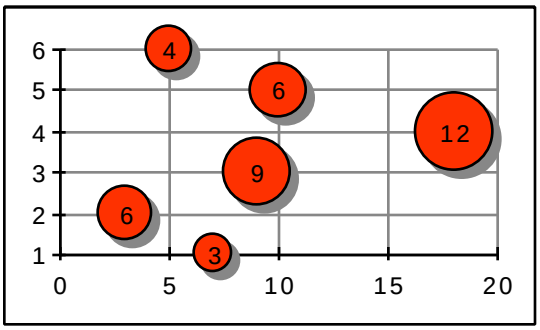
Die Funktion `BubbleChart()` dient zur Darstellung von eindimensionalen Blasendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die Positionen, die 2. Werteserie die Durchmesser zur ersten Serie, die 3. und 4. Werteserie in `ChartData()` die Positionen und Durchmesser zur zweiten Serie usw. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 4) // Durchmesser
     BubbleChart()
CloseDrawing()
```



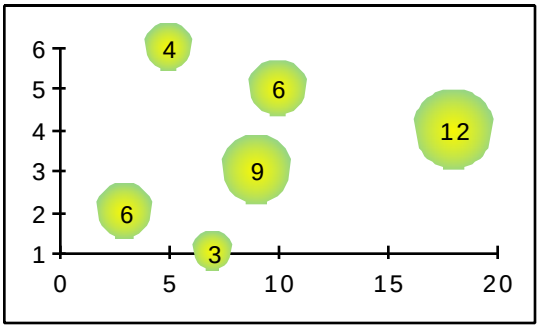
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Blasen mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 4) // Durchmesser
     BubbleChart(horizontal+shadow+label)
CloseDrawing()
```



Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiel:

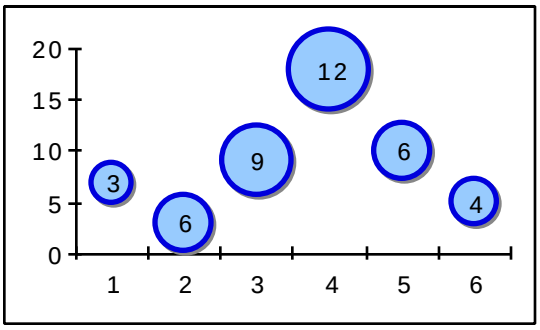
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 4) // Durchmesser
     BubbleChart(label+horizontal)
     PictureStyle(1;resource;"39")
     BorderStyle(1;none)
     GridLocation(all;none)
CloseDrawing()
```



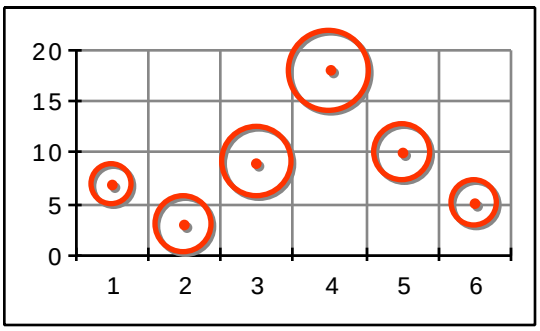
Bei eindimensionalen Blasendiagrammen besteht optional die Möglichkeit, alle Blasen um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 4) // Durchmesser
     BubbleChart(shadow+label;on)
     FillStyle(1;lightBlue)
     BorderStyle(1;;2;blue)
     ShadowStyle(all;1;gray)
     GridLocation(all;none)
CloseDrawing()
```



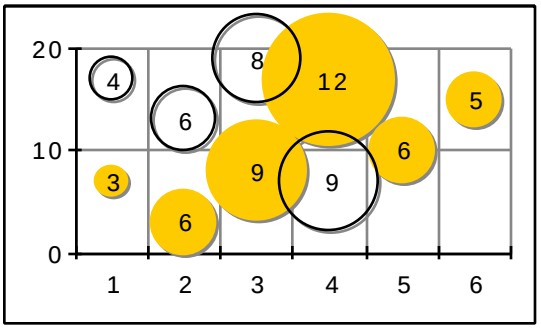

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 4) // Durchmesser
     BubbleChart(shadow+symbol;on)
     FillStyle(1;;transparent)
     BorderStyle(1;;2;red)
     SymbolStyle(1;bullet;3;;red)
     ShadowStyle(all;1;gray)
CloseDrawing()
```



BubbleChartOptions(maxDurchmesser;typ)

Mit dem ersten Argument *maxDurchmesser* kann der maximale Blasendurchmesser kontrolliert werden. Wird kein Wert vorgegeben, so ist standardmäßig der maximale Blasendurchmesser gleich 30 Pixel. Zusätzlich kann noch das Größenverhältnis der Blasen zueinander gesteuert werden, d.h. bei *typ=areaProp* verhalten sich die Flächen der Blasen zueinander proportional, bei *typ=diameterProp* verhalten sich die Durchmesser zueinander proportional. Standardmäßig ist *typ=areaProp*.
Beispiel:

```
        OpenDrawing(200;120)
        AddFrame(0;0;200;120)
        ChartData(17 13 19  7;       // Positionen  1.Serie
                  4  6  8  9;        // Durchmesser 1.Serie
                  7  3  8 17 10 15; // Positionen  2.Serie
                  3  6  9 12  6  5) // Durchmesser 2.Serie
        BubbleChart(shadow+label;on)
        BubbleChartOptions(50;diameterProp)
        FillStyle(1;;transparent)
        FillStyle(2;darkYellow)
        BorderStyle(2;none)
        ShadowStyle(all;1;gray)
CloseDrawing()
```



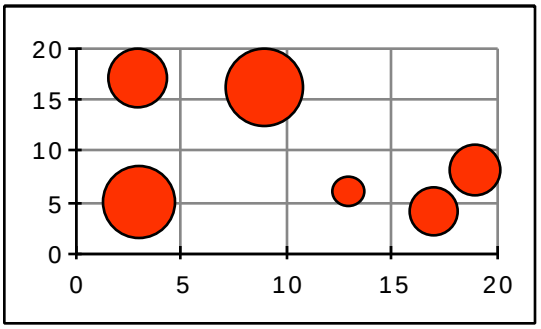
Zu beachten ist, dass die Funktion `BubbleChartOptions()` nach der Funktion `BubbleChart()` bzw. `BubbleChart2D()` anzuführen ist.

BubbleChart2D(darstellung)

Die Funktion `BubbleChart2D()` ermöglicht die Darstellung von zweidimensionalen Blasendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte und die 3. Werteserie die Durchmesser zur ersten Serie, die 4. und 5. Werteserie in `ChartData()` die x- und y-Werte zur zweiten Serie, die 6.

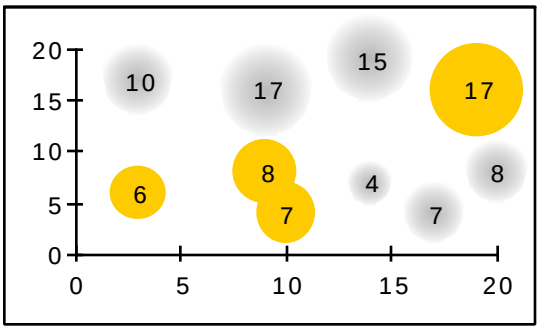
Werteserie die Durchmesser zur zweiten Serie usw. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(17 13 19   9  3  3; // x-Positionen
               4  6  8  16 17  5; // y-Positionen
               7  3  8  17 10 15) // Durchmesser
     BubbleChart2D()
CloseDrawing()
```



Wie bei den eindimensionalen Blasendiagrammen können auch hier durch das Argument *darstellung* die Blasen mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(17 14 20   9  3 14; // x-Positionen 1.Serie
               4  7  8  16 17 19; // y-Positionen 1.Serie
               7  4  8  17 10 15; // Durchmesser  1.Serie
              10  3  9  19;       // x-Positionen 2.Serie
               4  6  8  16;       // y-Positionen 2.Serie
               7  6  8  17)       // Durchmesser  2.Serie
     BubbleChart2D(label)
     BubbleChartOptions(35;areaProp)
     PictureStyle(1;resource;"33")
     FillStyle(2;darkYellow)
     BorderStyle(all;none)
     GridLocation(all;none)
CloseDrawing()
```

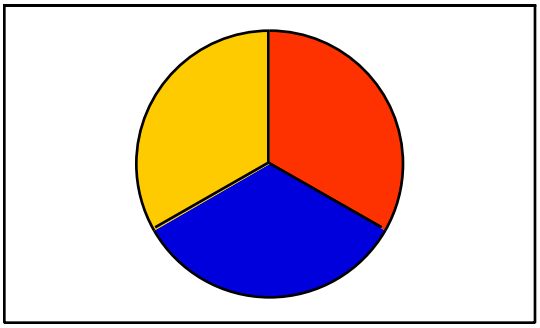


Kuchendiagramme:

PieChart(darstellung;3DHöhe;innenradius;startwinkel; öffnungswinkel)

Die Funktion `PieChart()` ermöglicht die Darstellung einer Vielzahl unterschiedlicher Tortendiagramme. Beispiel:

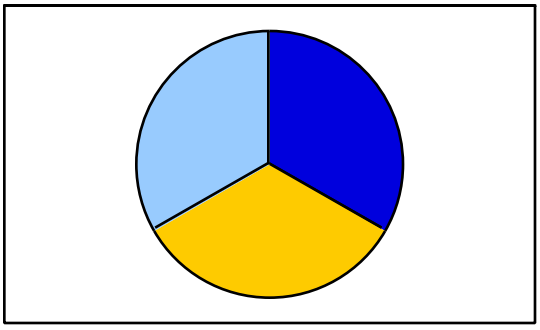
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(135 135 135)
     PieChart()
     FillStyle(3;darkYellow)
CloseDrawing()
```



Dargestellt wird nur die erste Werteserie in `ChartData()`, weitere Werteserien werden ignoriert. Das gleiche gilt für Werte kleiner gleich Null.

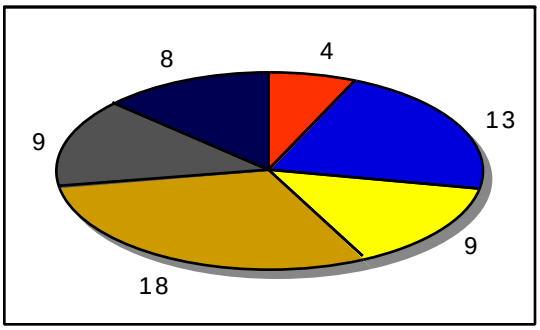
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     // Werte kleiner gleich Null werden übersprungen.
     ChartData(-100 135 135 0 135;
               100 200 300) // Weitere Serien ignoriert.
     PieChart()
     FillStyle(3;darkYellow)
     FillStyle(5;lightBlue)
CloseDrawing()
```



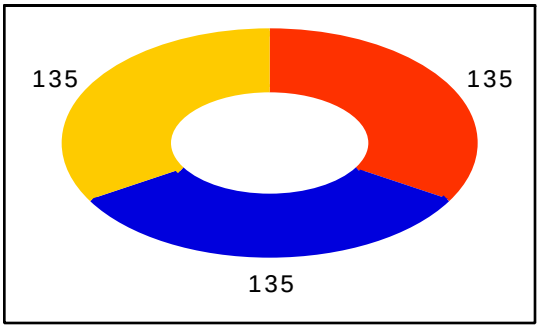
Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden alle Tortendiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13 9 18 9 8)
     PieChart(oval+label+shadow)
CloseDrawing()
```



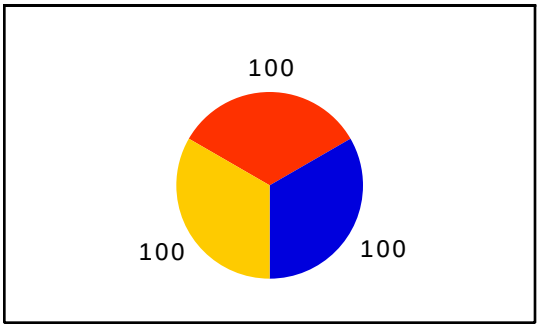
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden im Abschnitt *Stile* erläutert.
Das 2. Argument *3DHöhe* ist zur Zeit noch nicht implementiert. Das 3. Argument *innenradius* dient zur Darstellung von Ringdiagrammen. Der Innenradius ist dabei in Prozent der Segmentlänge einzugeben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(135 135 135)
     PieChart(label+oval;;50)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```



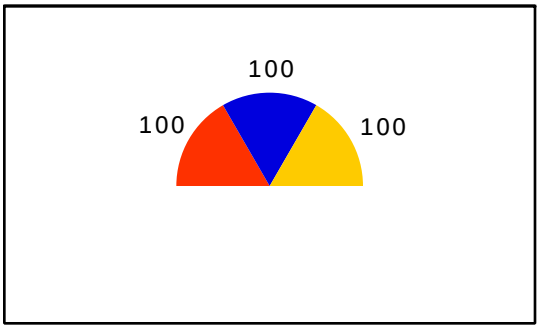
Standardmäßig werden die Segmente eines Tortendiagramms oben beginnend (12 Uhr), im Uhrzeigersinn über einen Bereich von 360 Grad (Vollkreis) aufgetragen. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position des ersten Segments als auch der Richtungsinn und der Öffnungswinkel des Kreissektors festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Segmente im Uhrzeigersinn, durch einen negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von ± 360 Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(100 100 100)
     PieChart(label;;;-60)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```

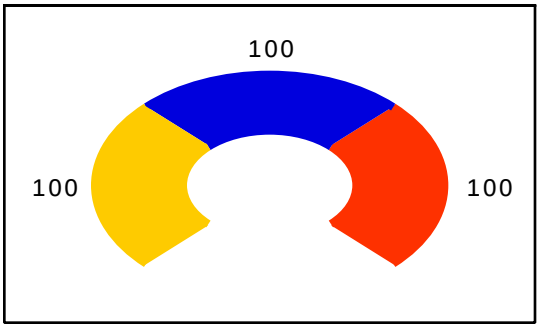


Bei einem Öffnungswinkel ungleich ±360 Grad werden die Segmente in Form eines Kreissektors angeordnet. Ein negativer Öffnungswinkel bewirkt, dass die Segmente gegen den Uhrzeigersinn aufgetragen werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(100 100 100)
     PieChart(label;;;-90;180)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```



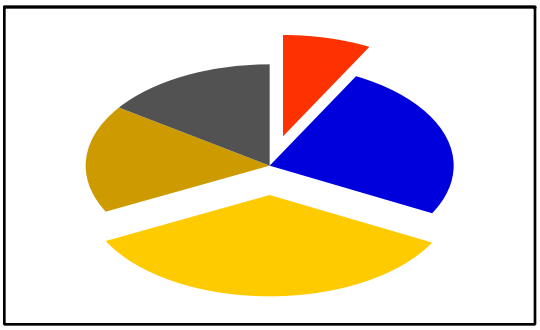

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(100 100 100)
     PieChart(oval+label;;50;135;-270)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```



PieChartExplodes(versatz;segment1;segment2...)

Durch die Funktion `PieChartExplodes()` können Segmente eines Tortendiagramms versetzt (explodierend) dargestellt werden. Das erste Argument *versatz* definiert die Größe des Versatzes, welcher in Prozent der Segmentlänge anzugeben ist. Falls kein Versatzwert vorgegeben wird, werden die Segmente um 20% der Segmentlänge versetzt (*versatz=20*). Die weiteren Argumente in `PieChartExplodes()` legen fest, welche Segmente vesetzt dargestellt werden sollen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13 18 9 8)
     PieChart(oval)
     PieChartExplodes(30;1;3)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```

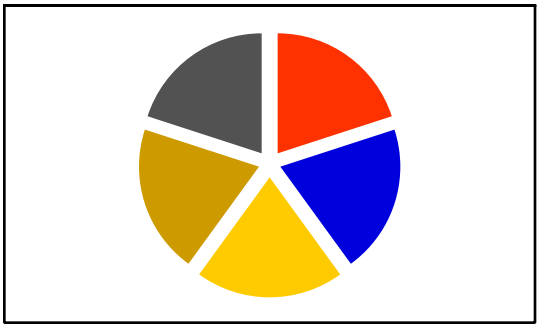


Zusätzlich stehen noch die folgenden Segmentkonstanten zur Verfügung:

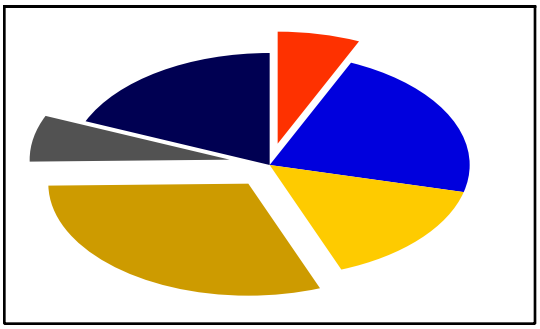
Konstante	*Versetzt darstellen*
`none`	kein Segment
`all`	alle Segmente
`max`	größtes Segment
`min`	kleinstes Segment

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(72 72 72 72 72)
     PieChart()
     PieChartExplodes(10;all)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13 9 18 4 11)
     PieChart(oval)
     PieChartExplodes(;max;min)
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
CloseDrawing()
```

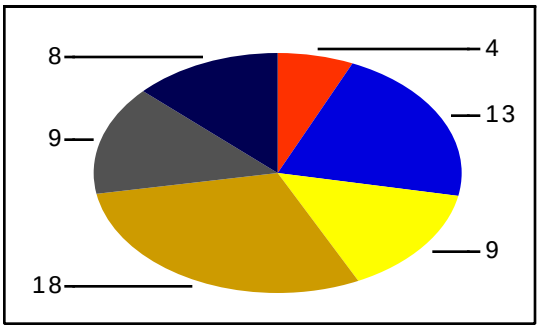


Zu beachten ist, dass die Funktion `PieChartExplodes()` nach der Funktion `PieChart()` anzuführen ist. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen.*

PieChartAuxLines(horizontaleLänge;extensionLänge; vAusrichtung;strichstärke;farbe;muster)

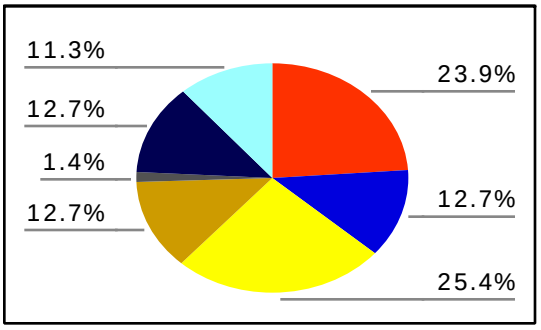
Die Funktion `PieChartAuxLines()` kann zur besseren Platzierung der außenliegenden Beschriftungen verwendet werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13 9 18 9 8)
     PieChart(label+oval)
     PieChartAuxLines()
     BorderStyle(all;none)
CloseDrawing()
```

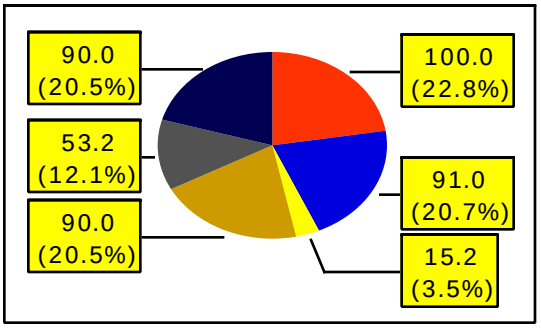


Die horizontale Länge der Hilfslinien kann mit dem Argument *horizontale-Länge* kontrolliert werden. Diese ist in Prozent der Segmentlänge anzugeben. Das Argument *extensionLänge* wird zur Zeit nicht genutzt und wird ignoriert. Mit dem Argument *vAusrichtung* kann die Position der Hilfslinie relativ zur Beschriftung festgelegt werden. Bei *vAusrichtung= bottom* wird die Hilfslinie unterhalb der Beschriftung platziert, bei *vAusrichtung= top* oberhalb, bei *vAusrichtung=center* in der Mitte (default). Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke, farbe* und *muster* variiert werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(17 9 18 9 1 9 8)
     PieChart(label+oval)
     PieChartAuxLines(;;bottom;1;gray)
     LabelTexts(;"|||f1|%")
     BorderStyle(all;none)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(100 91 15.2 90 53.2 89.97)
     PieChart(label+oval)
     PieChartAuxLines()
     LabelTexts(;"|f1|\n(|f1|%)")
     LabelStyle(;;;;;center)
     LabelBackground(1;yellow)
     BorderStyle(all;none)
CloseDrawing()
```



Die formatierte Ausgabe der Werte (absolut und/oder relativ) wird ausführlich inkl. Beispiele im Abschnitt *Stile* erläutert.

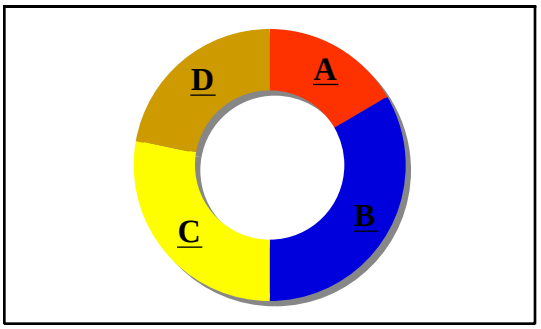
`PieChartLabelOptions(relativeLimitsVerwenden; textversatzAußen;textversatzInnen)`

Das 1. Argument *relativeLimitsVerwenden* ermöglicht anstelle absoluter Grenzwerte, relative Grenzwerte zu definieren: Diagrammwerte, welche nicht innerhalb dieser Limits liegen, werden nicht beschriftet. Die Grenzwerte, d.h. das untere und obere Limit werden in der Funktion `LabelOptions()` festgelegt.
Mit den Argumenten *textversatzAußen* und *textversatzInnen* kann der Abstand zwischen der Berandung und der äußeren bzw. inneren Beschriftung kontrolliert werden. Weitere Informationen dazu im Abschnitt *Stile*.

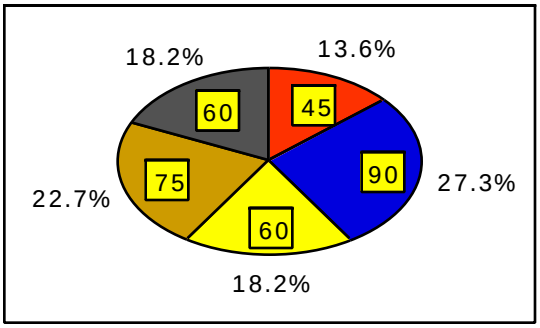
PieChartInnerLabelTexts(text1;text2...)
PieChartInnerLabelStyle(schrift;größe;stil;farbe;
ausrichtung)
PieChartInnerLabelBackground(füllfarbe;füllmuster;
rahmenbreite;rahmenfarbe;rahmenmuster;
schattenabstand;schattenfarbe;schattenmuster)

Standardmäßig wird die Beschriftung außerhalb des Tortendiagramms platziert (*darstellung=label*). Durch die Funktionen `PieChartInnerLabelTexts()`, `PieChartInnerLabelStyle()` und `PieChartInnerLabelBackground()` können zusätzlich auch innerhalb des Diagramms Beschriftungen angeordnet und gestaltet werden. Der Aufbau und die Wirkungsweise der drei Funktionen ist gleich der "Standard"-Funktionen `LabelTexts()`, `LabelStyle()` und `LabelBackground()`. Letztere werden detailliert im Abschnitt *Stile* behandelt. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 90 75 60)
     PieChart(shadow;;60)
     PieChartInnerLabelTexts("A";"B";"C";"D")
     PieChartInnerLabelStyle("Times";12;bold+underline)
     BorderStyle(all;none)
     ShadowStyle(all;2;gray)
CloseDrawing()
```



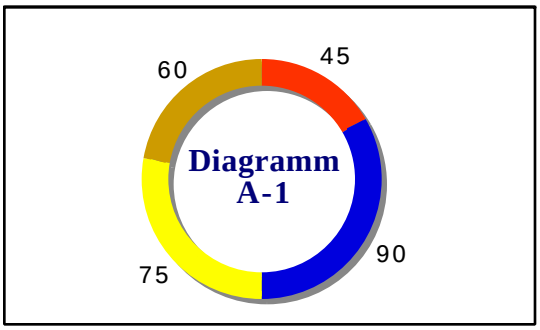

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 90 60 75 60)
     PieChart(label+oval)
     LabelTexts(1;"|||f1|%")        // relative Werte
     PieChartInnerLabelTexts("|u|") // absolute Werte
     PieChartInnerLabelBackground(yellow)
CloseDrawing()
```



PieChartCenterLabelText(text)
PieChartCenterLabelStyle(schrift;größe;stil;farbe;
ausrichtung)
PieChartCenterLabelBackground(füllfarbe;füllmuster;
rahmenbreite;rahmenfarbe;rahmenmuster;
schattenabstand;schattenfarbe;schattenmuster)

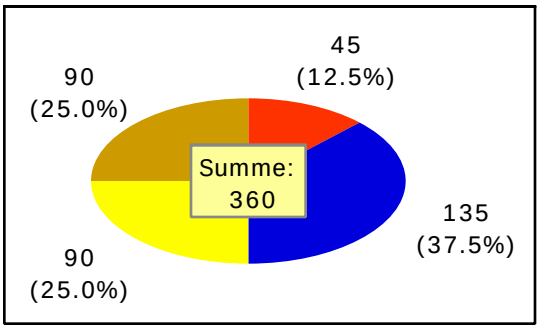
Zusätzlich kann durch die Funktionen `PieChartCenterLabelText()`, `PieChartCenterLabelStyle()` und `PieChartCenterLabelBackground()` ein Text in der Mitte des Diagramms angeordnet und gestaltet werden. Der Aufbau und die Wirkungsweise der drei Funktionen ist gleich der "Standard"-Funktionen `LabelTexts()`, `LabelStyle()` und `LabelBackground()`, siehe Abschnitt *Stile*. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 90 75 60)
     PieChart(shadow+label;;85)
     PieChartCenterLabelText("Diagramm\nA-1")
     PieChartCenterLabelStyle("Times";12;bold;darkBlue)
     BorderStyle(all;none)
     ShadowStyle(all;2;gray)
CloseDrawing()
```



Enthält der Text in der Funktion `PieChartCenterLabelText()` eine Formatanweisung, z.B. "|u|", so wird die Summe aller positiven Werte ausgegeben. Eine detaillierte Erklärung aller Formatanweisungen inkl. zahlreicher Beispiele ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 135 90 90)
     PieChart(oval+label)
     LabelTexts(1;"|u|\n(|f1|%)")
     LabelStyle(;;;;;center)
     PieChartCenterLabelText("Summe: \n|u|")
     PieChartCenterLabelBackground(lightYellow;;1;gray)
     BorderStyle(all;none)
CloseDrawing()
```

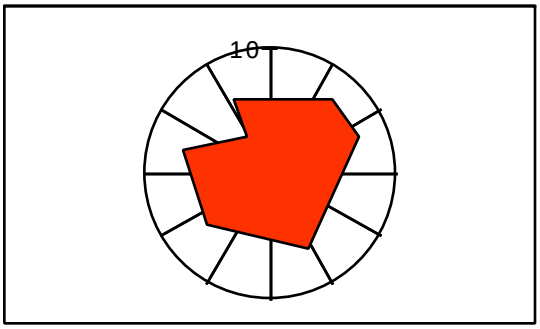


Polardiagramme:

PolarChart(darstellung;startwinkel;öffnungswinkel)

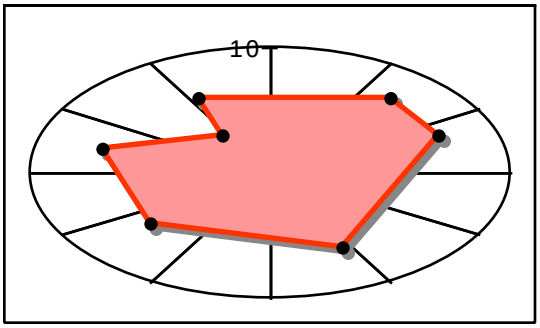
Die Funktion `PolarChart()` ermöglicht die Darstellung von zweidimensionalen Werten in Polarkoordinaten. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte zur ersten Serie, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte zur zweiten Serie usw. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5  7  3 -5 -7 -2 -3; // x-Werte
               6  3 -6 -4  2  3  6) // y-Werte
     PolarChart()
     MajorGridLineColors(all;all;black)
CloseDrawing()
```

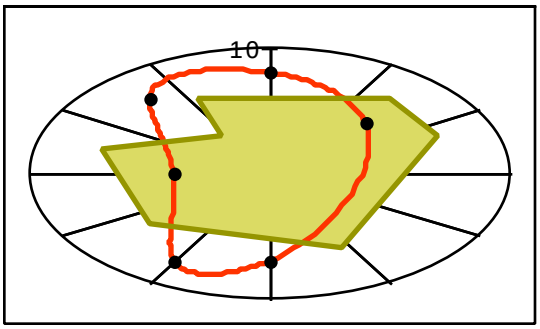


Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden Polardiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval+shadow+symbol)
     FillStyle(1;lightRed)
     BorderStyle(1;poly;2;red)
     SymbolStyle(1;bullet;4;1;black)
     ShadowStyle(1;2;gray)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```

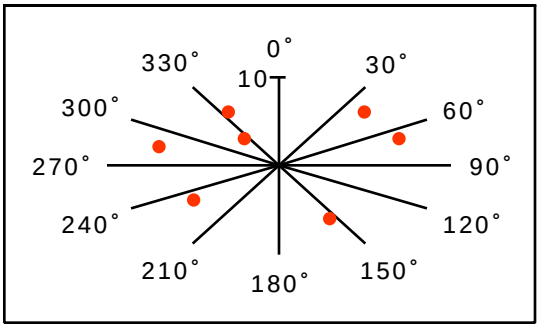


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5  7  3 -5 -7 -2 -3; // x-Werte 1.Serie
               6  3 -6 -4  2  3  6; // y-Werte 1.Serie
               0  4  0 -4 -4 -5;    // x-Werte 2.Serie
               8  4 -7 -7  0  6)    // y-Werte 2.Serie
     PolarChart(oval+symbol)
     // 1. Serie
     FillStyle(1;220 220 100)
     BorderStyle(1;poly;2;150 150 0)
     SymbolStyle(1;none)
     // 2. Serie
     FillStyle(2;;transparent)
     BorderStyle(2;smooth;2;red)
     SymbolStyle(2;bullet;4;1;black)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```



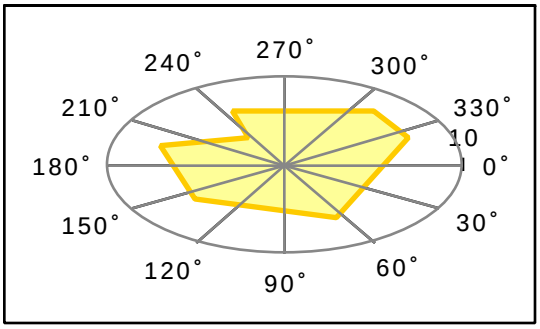
Standardmäßig werden die Punkte einer Serie durch eine polygonal begrenzte Fläche dargestellt. Durch Ausblenden sowohl der Füllung als auch der Berandung mittels der Funktionen `FillStyle(;;transparent)` und `BorderStyle(;none)` ist es möglich, die Diagrammwerte ausschließlich durch Symbole darzustellen. Beispiel:

```
OpenDrawing(200;120)
   AddFrame(0;0;200;120)
   ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
             6 3 -6 -4  2  3  6) // y-Werte
   PolarChart(oval+symbol)
   FillStyle(;;transparent)
   BorderStyle(;none)
   SymbolStyle(1;bullet;4;1;red)
   AxisLabelText(all;"|u|°")
   GridLocation(all;none) // kein Raster
CloseDrawing()
```

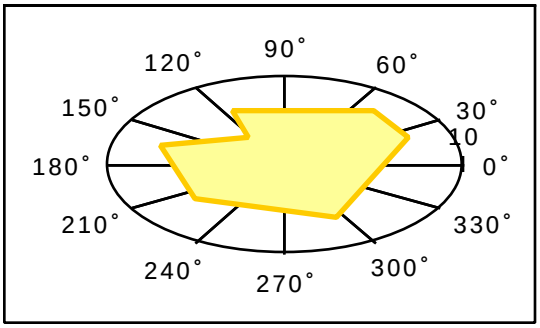


Die Texte der Achsenbeschriftungen werden mit der Funktion `AxisLabelText()` definiert; Details dazu finden sich im Abschnitt *Achsen*. Standardmäßig beginnt die Achsenbeschriftung oben (12 Uhr) und verläuft im Uhrzeigersinn über einen Bereich von 360 Grad. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position der "Nulllinie" als auch der Richtungsinn und der Öffnungswinkel des Rasters festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Achsen im Uhrzeigersinn, durch einen negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von ±360 Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     // 90...0-Grad-Linie liegt "rechts"
     PolarChart(oval;90)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     GridLocation(all;front)
     AxisLabelText(all;"|u|˚")
CloseDrawing()
```



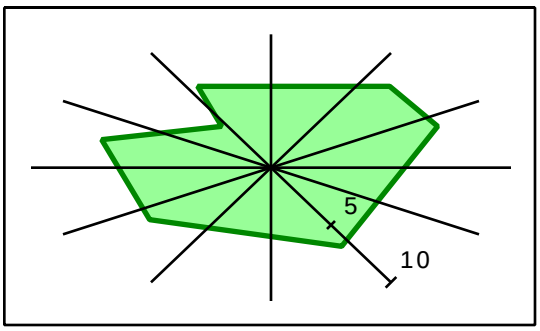
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     //   90...0-Grad-Linie liegt "rechts"
     // -360...Skalierung gegen den Uhrzeigersinn
     PolarChart(oval;90;-360)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     MajorGridLineColors(all;all;black)
     AxisLabelText(all;"|u|°")
CloseDrawing()
```



PolarChartOptions(skalenachse;rasterform;
pfeileHinzufügen;polygonNichtSchließen;achsenanzahl)

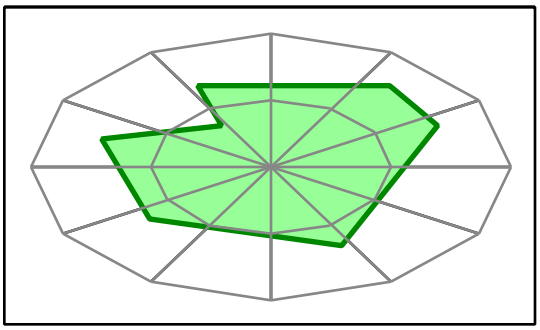
Polardiagramme können durch die Funktion `PolarChartOptions()` auf vielfältige Weise variiert werden. Zu beachten ist, dass die Funktion `PolarChartOptions()` nach der Funktion `PolarChart()` anzuführen ist. Mit dem ersten Argument *skalenachse* kann festgelegt werden, entlang welcher Achse die Skalierung aufgetragen wird. Standardmäßig wird die Skalierung der ersten Achse zugeordnet (*skalenachse=1*), dies ist die Achse bei Null Grad. Bei *skalenachse=0* wird keine Skala ausgegeben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval)
     PolarChartOptions(6) // nach PolarChart()!
     AxisOptions(all;front)
     FillStyle(1;lightGreen)
     BorderStyle(1;poly;2;green)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



Mit dem zweiten Argument *rasterform* kann zwischen dem standardmäßigen ovalen Raster (*rasterform=oval*), einem polygonalen Raster (*rasterform=poly*) und einem ausgeblendeten Raster (*rasterform=none*) gewählt werden. Das Raster kann auch durch die Funktion `GridLocation(;none)` ausgeblendet werden. Beispiel:

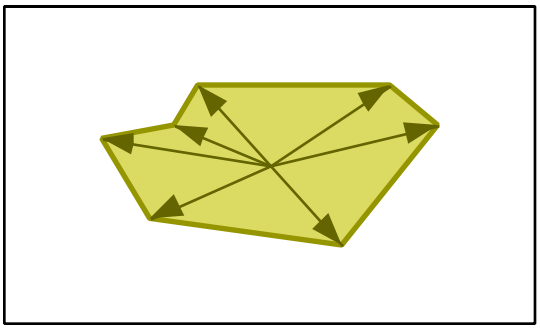
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval)
     PolarChartOptions(0;poly) // nach PolarChart()!
     FillStyle(1;lightGreen)
     BorderStyle(1;poly;2;green)
     GridLocation(all;front)
CloseDrawing()
```



Optional können Polardiagrammen durch vom Zentrum ausgehende Pfeile ergänzt werden (*pfeileHinzufügen=on*). Die Gestaltung der Pfeile erfolgt durch die Funktion `ArrowStyle()` und wird im Abschnitt *Stile* behandelt.

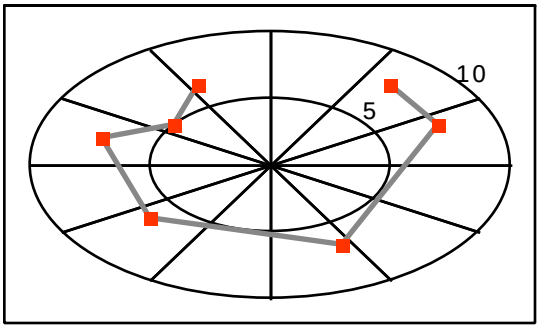
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -4 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval)
     PolarChartOptions(0;none;on) // nach PolarChart()!
     FillStyle(1;220 220 100)
     BorderStyle(1;poly;2;150 150 0)
     ArrowStyle(1;1;100 100 0;;;12;8)
     AxisOptions(all;none) // keine Achsen
CloseDrawing()
```



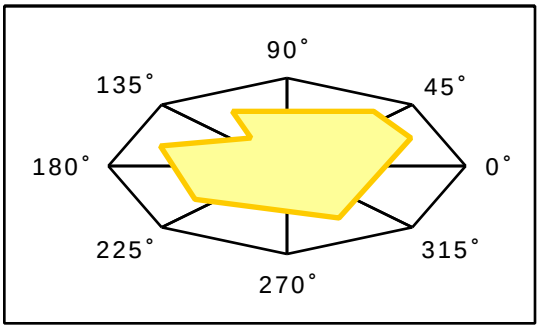
Durch die Funktion `FillStyle(;;transparent)` kann die Füllung ausgeblendet werden; die so verbleibende Berandung verbindet die Punkte in Form eines geschlossenen Linienzuges. Durch Aktivieren des 4. Argumentes *polygonNichtSchließen=on* kann die Schlusslinie zwischen dem letzten und ersten Punkt ausgeblendet werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -4 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval+symbol)
     PolarChartOptions(3;oval;off;on) // nach PolarChart()
     FillStyle(1;;transparent)
     BorderStyle(1;poly;2;gray)
     SymbolStyle(1;square;4;1;red)
     AxisMajorTicks(1;0)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```



Die Anzahl der radialen Rasterlinien (Achsen) kann mit dem 5. Argument *achsenanzahl* kontrolliert werden. Standardmäßig werden Polardiagramme mit 12 radialen Rasterlinien dargestellt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     //   90...0-Grad-Linie liegt "rechts"
     // -360...Skalierung gegen den Uhrzeigersinn
     PolarChart(oval;90;-360)
     PolarChartOptions(0;poly;off;;8) // nach PolarChart()
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     MajorGridLineColors(all;all;black)
     AxisLabelText(all;"|u|°")
CloseDrawing()
```

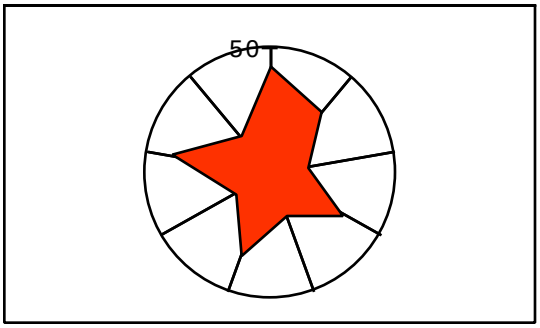


Radardiagramme:

RadarChart(darstellung;startwinkel;öffnungswinkel)

Bei der Funktion `RadarChart()` werden die Diagrammwerte entlang strahlenförmig ausgerichteter Achsen angeordnet. Die Anzahl der Achsen wird durch die Anzahl der Werten bestimmt. Die einzelnen Werte einer Serie werden standardmäßig zu einer polygonal begrenzten Fläche verbunden. Beispiel:

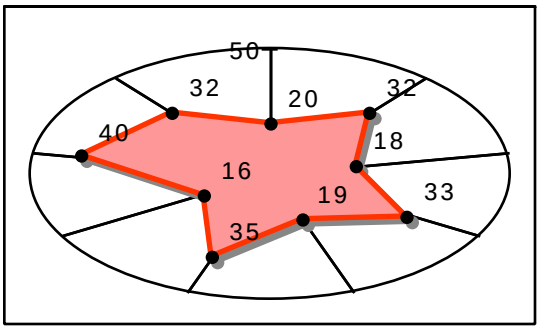
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(43 32 15 33 18 35 16 40 19)
     RadarChart()
     MajorGridLineColors(all;all;black)
CloseDrawing()
```



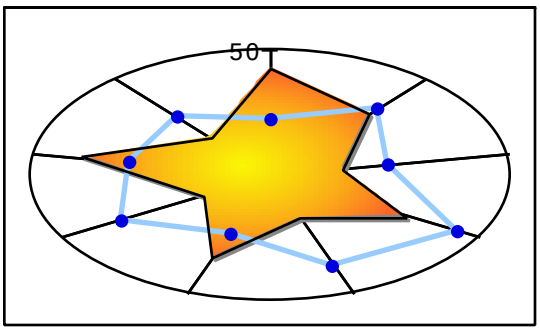
Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden Radardiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(20 32 18 33 19 35 16 40 32)
     RadarChart(oval+symbol+label+shadow)
     FillStyle(1;lightRed)
     BorderStyle(1;poly;2;red)
     ShadowStyle(1;2;gray)
     SymbolStyle(1;bullet;4;;black)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```

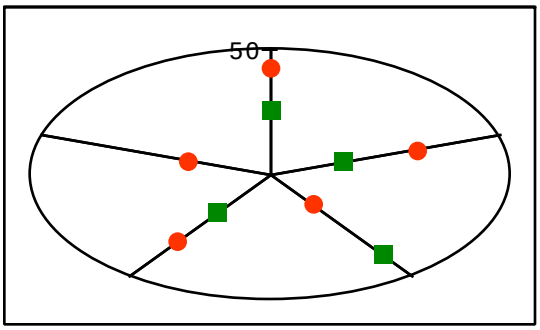



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(43 32 15 33 18 35 16 40 19; // 1.Serie
               22 35 25 45 38 25 36 30 30) // 2.Serie
     RadarChart(oval+symbol+shadow)
     // 1. Serie
     PictureStyle(1;resource;"41")
     SymbolStyle(1;none)
     ShadowStyle(1;1;gray)

     // 2. Serie
     FillStyle(2;;transparent)
     BorderStyle(2;poly;2;lightBlue)
     SymbolStyle(2;bullet;4;1;)
     ShadowStyle(2;0)

     MajorGridLineColors(all;all;black)
CloseDrawing()
```



Durch Ausblenden der Füllung und der Berandung ist es möglich, die Diagrammwerte ausschließlich durch Symbole darzustellen. Beispiel:

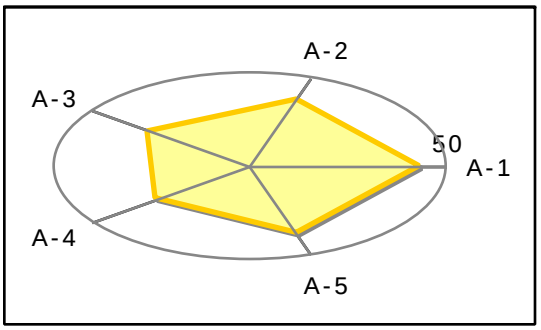
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(43 32 15 33 18; // 1. Serie
               25 16 40 19)    // 2. Serie
     RadarChart(oval+symbol)
     FillStyle(;;transparent)
     BorderStyle(;none)
     SymbolStyle(1;bullet;6;1;red)
     SymbolStyle(2;square;6;1;green)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```



Die Texte der Achsenbeschriftungen werden mit der Funktion `AxisLabelText()` definiert, Details dazu finden sich im Abschnitt *Achsen*. Der Achsenindex ist bei Radardiagrammen stets 1, Achsenindizes größer 1 werden ignoriert. Standardmäßig beginnt die Achsenbeschriftung oben (12 Uhr) und verläuft im Uhrzeigersinn über einen Bereich von 360 Grad. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position der ersten Achse als auch der Richtungsinn und der Öffnungswinkel der Achsen festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Achsen im Uhrzeigersinn, durch einen

negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von ± 360 Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(43 38 32 30 38)
     RadarChart(oval+shadow;90;-360)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     ShadowStyle(1;1;gray)
     GridLocation(all;front)
     AxisMajorTicks(;0)
     AxisLabelText(all;"A-|u|")
CloseDrawing()
```



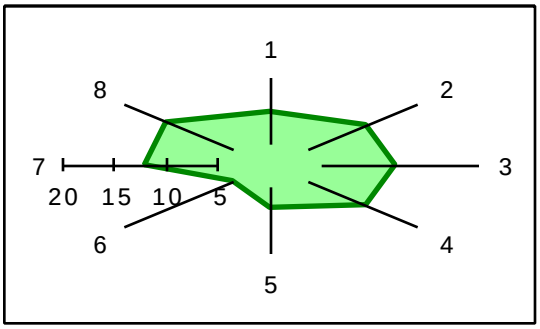
RadarChartOptions(skalenachse;rasterform;
pfeileHinzufügen;polygonNichtSchließen)

Radardiagramme können durch die Funktion `RadarChartOptions()` auf vielfältige Weise variiert werden. Zu beachten ist, dass die Funktion `RadarChartOptions()` nach der Funktion `RadarChart()` anzuführen ist.

Mit dem ersten Argument *skalenachse* kann festgelegt werden, entlang welcher Achse die Skalierung aufgetragen wird. Standardmäßig wird die Skalierung der ersten Achse zugeordnet (*skalenachse=1*). Bei *skalenachse=0* wird keine Skala ausgegeben.

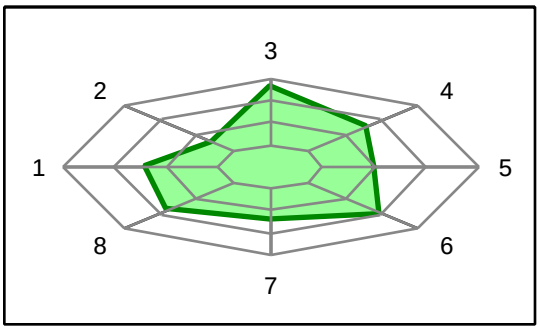
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 13 12 13 10 5 12 14)
     RadarChart(oval)
     RadarChartOptions(7) // nach RadarChart()!
     AxisOptions(all;front)
     FillStyle(1;lightGreen)
     BorderStyle(1;poly;2;green)
     GridLocation(all;none)
     Scaling(x;linear;5;20;3)
     AxisLabelText(all;"|u|")
CloseDrawing()
```

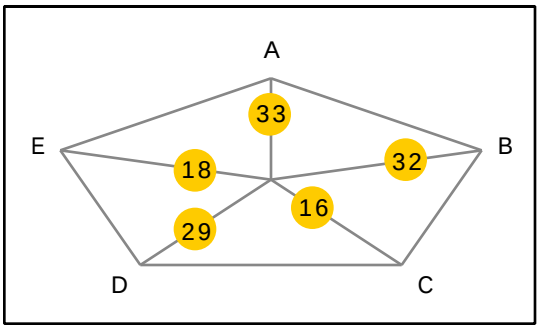


Mit dem zweiten Argument *rasterform* kann zwischen dem standardmäßigen ovalen Raster (*rasterform=oval*), einem polygonalen Raster (*rasterform=poly*) und einem ausgeblendeten Raster (*rasterform=none*) gewählt werden. Das Raster kann auch durch die Funktion `GridLocation(;none)` ausgeblendet werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 8 18 13 10 15 12 14)
     RadarChart(oval;-90)
     RadarChartOptions(0;poly) // nach RadarChart()!
     FillStyle(1;lightGreen)
     BorderStyle(1;poly;2;green)
     GridLocation(all;front)
     Scaling(x;linear;5;20;3)
     AxisLabelText(all;"|u|")
CloseDrawing()
```



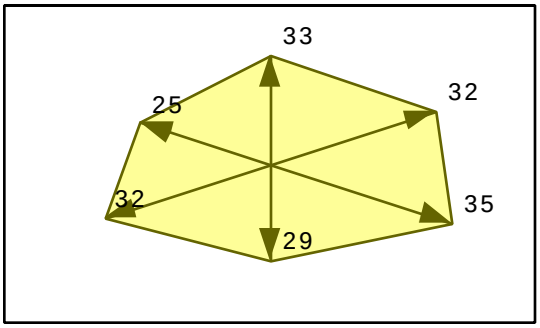

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(33 32 16 29 18)
     RadarChart(oval+label+symbol)
     RadarChartOptions(0;poly) // nach RadarChart()!
     FillStyle(1;;transparent)
     BorderStyle(1;none)
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(1;centerCenter)
     AxisLine(1;0) // keine Achsenlinien
     AxisLabelText(1;"A";"B";"C";"D";"E")
CloseDrawing()
```



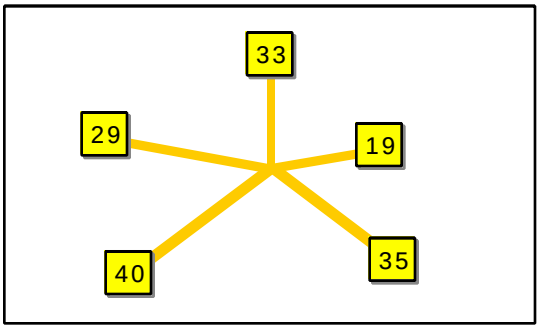
Optional können Radardiagrammen durch vom Zentrum ausgehende Pfeile ergänzt werden (*pfeileHinzufügen=on*). Die Gestaltung der Pfeile erfolgt durch die Funktion `ArrowStyle()` und wird im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(33 32 35 29 32 25)
     RadarChart(oval+label)
     RadarChartOptions(0;none;on) // nach RadarChart()!
     FillStyle(1;lightYellow)
     BorderStyle(1;;1;100 100 0)
     ArrowStyle(1;1;100 100 0;;;12;8)
     AxisOptions(all;none) // keine Achsen
CloseDrawing()
```

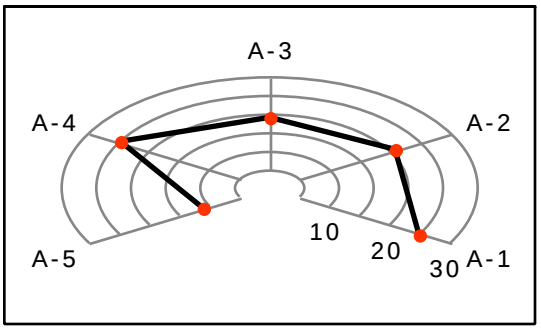



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(33 19 35 40 29)
     RadarChart(oval+label)
     RadarChartOptions(0;none;on) // nach RadarChart()!
     FillStyle(1;;transparent)
     BorderStyle(1;none)
     LabelBackground(1;yellow;;1;black;;1)
     LabelOptions(1;centerCenter)
     ArrowStyle(1;3;darkYellow;;;0;0)
     AxisOptions(all;none) // keine Achsen
CloseDrawing()
```

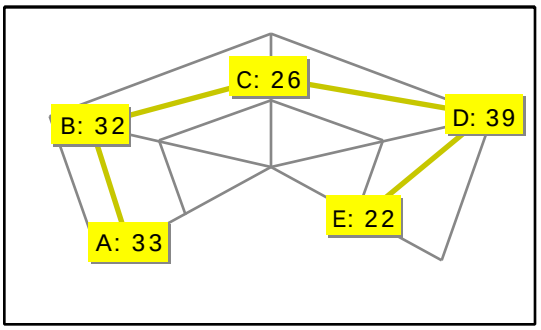


Durch die Funktion `FillStyle(;;transparent)` kann die Füllung ausgeblendet werden, die so verbleibende Berandung verbindet die Punkte in Form eines geschlossenen Linienzuges. Durch Aktivieren des 4. Arguments *polygonNichtSchließen=on* kann die Schlusslinie zwischen dem letzten und ersten Punkt ausgeblendet werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(25 21 19 25 11)
     RadarChart(symbol+oval;120;-240)
     RadarChartOptions(1;oval;off;on) // nach RadarChart()

     FillStyle(all;;transparent)
     BorderStyle(1;poly;2;black)
     SymbolStyle(1;bullet;4;1;red)

     Scaling(1;linear;5;30;5)
     AxisLabelText(1;"A-|i0|")
     AxisLine(1;0)       // keine Achsenlinien
     AxisMajorTicks(1;0) // keine Skalen-Markierungen
     AxisMajorTickLabelOptions(1;in;;;2;2)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(33 32 26 39 22)
     RadarChart(oval+label;-135;270)
     RadarChartOptions(0;poly;off;on) // nach RadarChart()

     LabelTexts(1;"A: |u|";"B: |u|";"C: |u|";
                 "D: |u|";"E: |u|")
     LabelBackground(1;yellow;;0;;;1)
     LabelOptions(1;centerCenter)

     FillStyle(1;;transparent)
     BorderStyle(1;;2;200 200 0)

     AxisOptions(all;none) // keine Achsen
CloseDrawing()
```



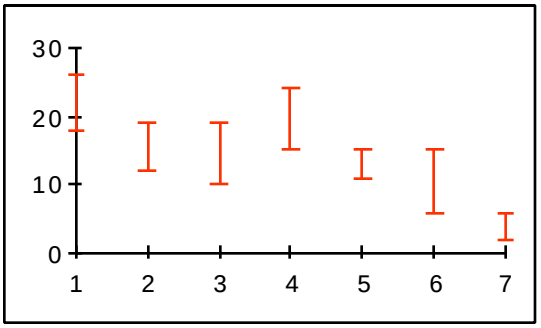
HochTief-Diagramme:

HighLowChart(darstellung;intervalleVersch.;diagrammTyp;
hochMarkerLänge;hochMarkerAusrichtung;
tiefMarkerLänge;tiefMarkerAusrichtung;
schlussMarkerLänge;schlussMarkerAusrichtung;
öffnenMarkerLänge;öffnenMarkerAusrichtung)

Die Funktion `HighLowChart()` ermöglicht die vielfältige Darstellung von Wertebereichen. Zur Festlegung der Bereichsgrenzen sind mindestens zwei Werteserien notwendig. Die Höchstwerte werden in der Funktion `ChartData()` als 1. Werteserie, die Tiefstwerte als 2. Werteserie angegeben.

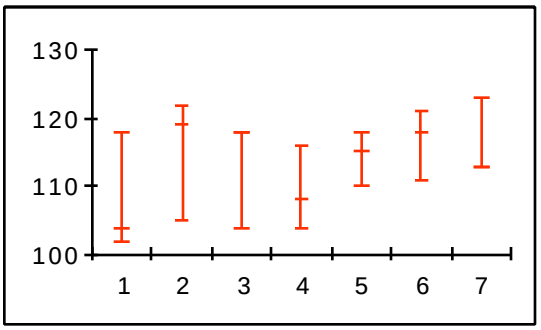
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(26 19 19 24 15 15 6; // Höchstwerte
               18 12 10 15 11  6 2) // Tiefstwerte
     HighLowChart()
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



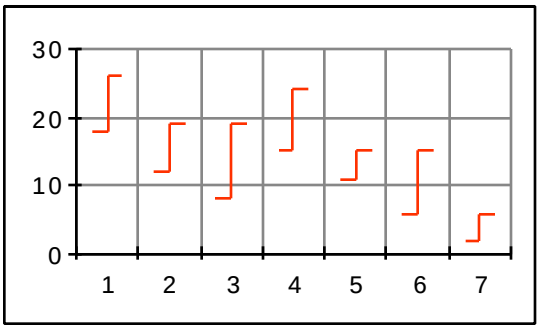
Optional können unter Verwendung des 3. Arguments *diagrammTyp*, neben den standardmäßig verwendeten zwei Werten (*diagrammTyp=highLow*), auch drei Werte (*diagrammTyp=highLowClose*) oder auch vier Werte (*diagrammTyp=highLowCloseOpen*) angegeben werden. Da diese Art der Darstellung häufig für Börsenkurse verwendet wird, werden die vier Werte als Höchst-, Tiefst-, Schluss- und Eröffnungskurs bezeichnet. Die Schlusskurse werden in `ChartData()` als 3. Werteserie, die Eröffnungskurse als 4. Werteserie angeführt.
Weiters besteht die Möglichkeit, die Bereichslinien um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(118 122 118 116 118 121 123; //Höchstwerte
               102 105 104 104 110 111 113; //Tiefstwerte
               104 119 118 108 115 118 113) //Schlusswerte
     HighLowChart(;on;highLowClose)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

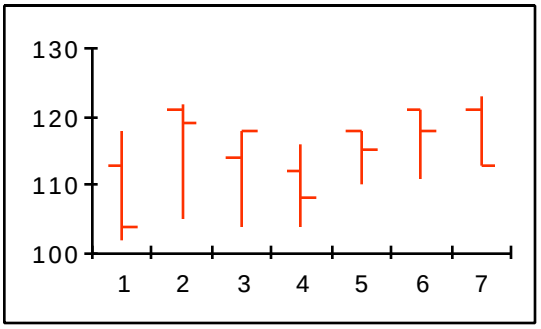


Die Ausrichtung der Markierungen — die Markierungen für die Tiefst- und Eröffnungskurse werden üblicherweise nach links ausgerichtet, die Markierungen für die Höchst- und Schlusskurse nach rechts — wird mit den Argumenten *hochMarkerAusrichtung*, *tiefMarkerAusrichtung*, *schlussMarkerAusrichtung* und *öffnenMarkerAusrichtung* festgelegt. Alle Markierungen können wahlweise links, rechts oder mittig (default) angeordnet werden. Die Länge der Markierungen kann mit den Argumenten *hochMarkerLänge*, *tiefMarkerLänge*, *schlussMarkerLänge* und *öffnenMarkerLänge* kontrolliert werden und ist in Prozent der Intervallbreite einzugeben. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(26 19 19 24 15 15 6; // Höchstwerte
               18 12  8 15 11  6 2) // Tiefstwerte
     HighLowChart(;on;highLow;;right;;left)
CloseDrawing()
```

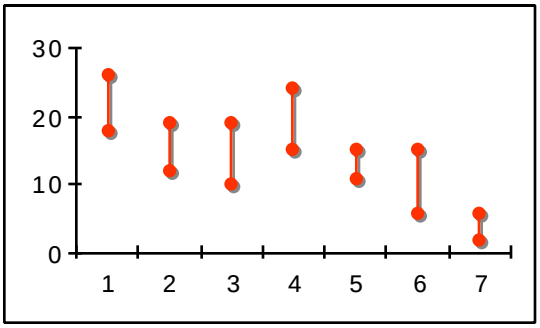


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(118 122 118 116 118 121 123; //Höchstwerte
               102 105 104 104 110 111 113; //Tiefstwerte
               104 119 118 108 115 118 113; //Schlusswerte
               113 121 114 112 118 121 121) //Eröffnungsw.
     HighLowChart(;on;highLowCloseOpen;0;;0;;;right;;left)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

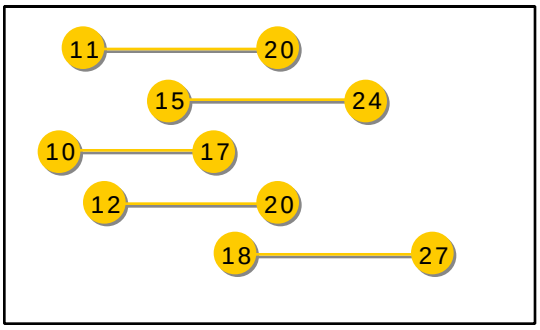


Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.
Die Darstellung der Linien, Symbole und Schatten kann durch die Stilfunktionen `LineStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* behandelt. Beispiele:

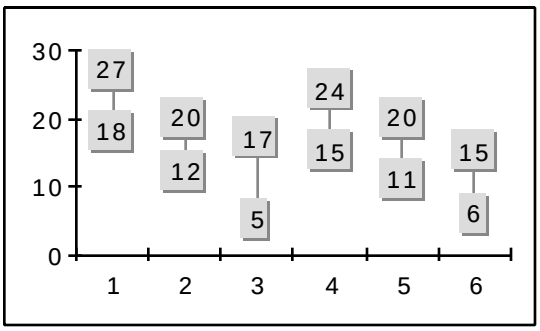
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(26 19 19 24 15 15 6; // Höchstwerte
               18 12 10 15 11  6 2) // Tiefstwerte
     HighLowChart(shadow+symbol;on;;0;;0)
     GridLocation(all;none) // kein Raster
     SymbolStyle(1;bullet;4)
     ShadowStyle(1;1;gray)
CloseDrawing()
```



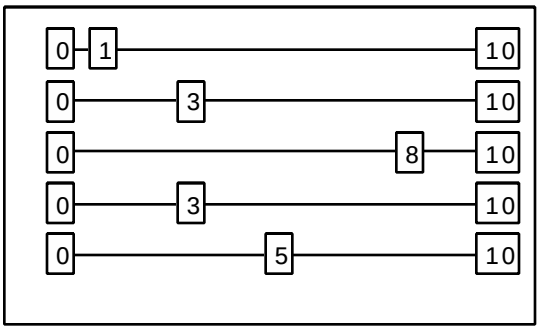

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(27 20 17 24 20; // Höchstwerte
               18 12 10 15 11) // Tiefstwerte
     HighLowChart(horizontal+shadow+label+symbol)
     LineStyle(1;poly;1;darkYellow)
     SymbolStyle(1;bullet;15;;darkYellow)
     ShadowStyle(1;1;gray)
     LabelOptions(1;centerCenter)
     GridLocation(all;none) // kein Raster
     AxisOptions(all;none)  // keine Achsen
CloseDrawing()
```



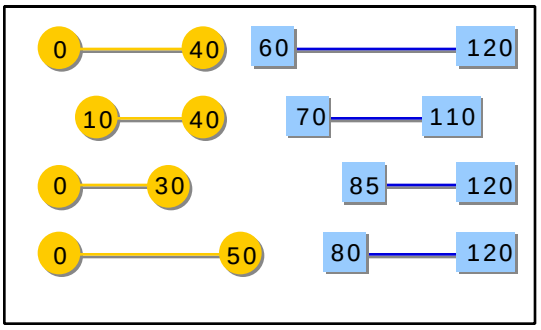

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(27 20 17 24 20 15; // Höchstwerte
               18 12  5 15 11  6) // Tiefstwerte
     HighLowChart(label;on)
     GridLocation(all;none) // kein Raster
     LineStyle(1;poly;1;gray)
     LabelBackground(all;lightGray;;0;;;1)
     LabelOptions(all;centerCenter)
CloseDrawing()
```



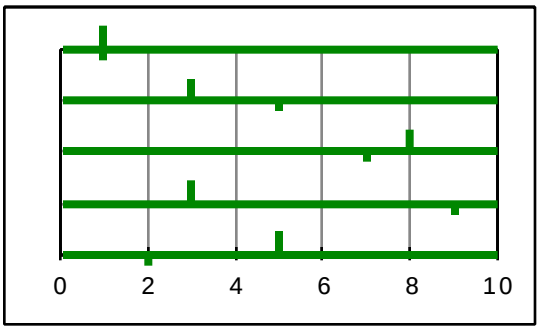

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(10 10 10 10 10; // Höchstwerte
               0  0  0  0  0; // Tiefstwerte
               5  3  8  3  1) // "Schlusswerte"
     HighLowChart(label+horizontal;;highLowClose)
     LineStyle(1;poly;1;black)
     LabelBackground()
     LabelOptions(all;centerCenter)
     GridLocation(all;none) // kein Raster
     AxisOptions(all;none)  // keine Achsen
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(50  30  40  40; // Max. 1.Serie
                0   0  10   0; // Min. 1.Serie
              120 120 110 120; // Max. 2.Serie
               80  85  70  60) // Min. 2.Serie
     HighLowChart(label+horizontal+symbol+shadow;;;0;;0)
     LineStyle(1;poly;1;darkYellow)
     SymbolStyle(1;bullet;15;;darkYellow)
     ShadowStyle(all;1;gray)
     SymbolStyle(2;none)
     LabelBackground(2;lightBlue;;0;;;1)
     LabelOptions(all;centerCenter)
     GridLocation(all;none) // kein Raster
     AxisOptions(all;none)  // keine Achsen
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(10 10 10 10 10; // Höchstwerte
                0  0  0  0  0; // Tiefstwerte
                5  3  8  3  1; // "Schlusswerte"
                2  9  7  5  1) // "Eröffnungswerte"
     HighLowChart(horizontal;off;highLowCloseOpen;
                              0;;       // Max.
                              0;;       // Min.
                              50;left;  // Schluss
                              20;right) // Öffnen
     AxisOptions(y;none)  // keine y-Achse
     LineStyle(1;poly;3;green)
     GridFrame()
CloseDrawing()
```

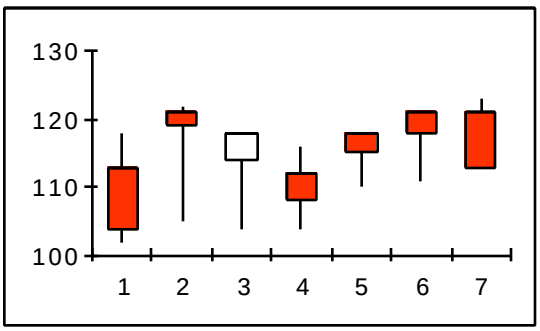


Candlestick-Diagramme:

**CandlestickChart(darstellung;abstand;hochMarkerLänge;
hochMarkerAusrichtung;tiefMarkerLänge;
tiefMarkerAusrichtung)**

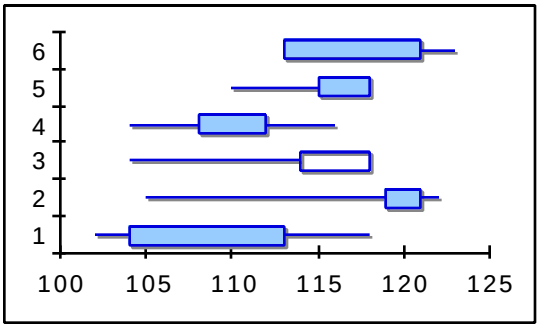
Die Funktion `CandlestickChart()` wird zur Darstellung von Börsenkursen verwendet. Die dazu in der Funktion `ChartData()` benötigten vier Werteserien entsprechen den Höchst-, Tiefst-, Schluss- und Eröffnungswerten. Dabei wird der höchste und der tiefste Kurswert durch eine Linie verbunden und der Bereich zwischen Eröffnungs- und Schlusskurs durch ein Rechteck dargestellt. Ist der Schlusskurs niedriger als der Eröffnungskurs, so wird die Rechteckfläche gefüllt dargestellt, umgekehrt, ist der Schlusskurs höher als der Eröffnungskurs, so wird die Rechteckfläche nicht gefüllt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(118 122 118 116 118 121 123; //Höchstwerte
               102 105 104 104 110 111 113; //Tiefstwerte
               104 119 118 108 115 118 113; //Schlusswerte
               113 121 114 112 118 121 121) //Eröffnungsw.
     CandlestickChart()
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*) und andererseits die Ergänzung mit einem Schatten (*darstellung=shadow*). Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Die Darstellung der Rechteckfüllungen, der Berandungen und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()` und `ShadowStyle()` variiert werden. Alle Stilfunktionen werden im Abschnitt *Stile* behandelt. Beispiele:

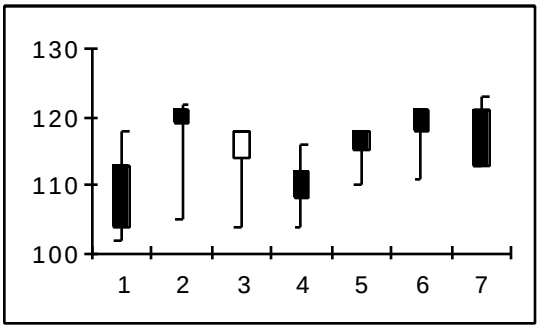
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(118 122 118 116 118 123; // Höchstwerte
               102 105 104 104 110 113; // Tiefstwerte
               104 119 118 108 115 113; // Schlusswerte
               113 121 114 112 118 121) // Eröffnungswerte
     CandlestickChart(shadow+horizontal)
     FillStyle(1;lightBlue)
     BorderStyle(1;;1;blue)
     ShadowStyle(1;1;gray)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



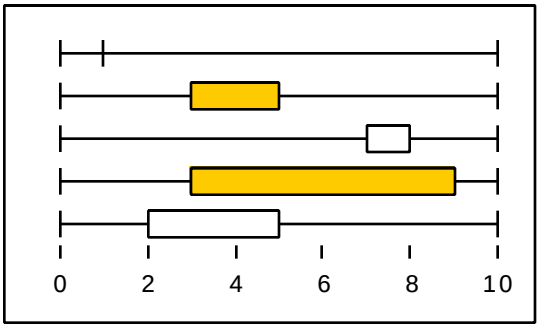
Mit dem 2. Argument *abstand* kann die Distanz zwischen den einzelnen "Candlesticks" bzw. die Breite der Rechtecke kontrolliert werden. Der Abstand wird dabei in Prozent der Rechteckbreite angegeben. Standardmäßig ist der Abstand gleich der Rechteckbreite (*abstand=100*). Bei Werten größer 100 (z.B. *abstand=400*) werden die Rechtecke schmäler und die Abstände dazwischen größer, bei einem Wert kleiner 100 (z.B. *abstand=50*) werden die Rechtecke breiter und die Äbstande kleiner.
Optional können die "Candlesticks" durch links, rechts oder mittig ausgerichtete Markierungen ergänzt werden. Mit dem 4. und 5. Argument *hochMarkerLänge* und *hochMarkerAusrichtung* werden die Länge und Ausrichtung der Markierungen für die Höchstwerte, mit dem 6. und 7. Argument *tiefMarkerLänge* und *tiefMarkerAusrichtung* die Länge und Aus-

richtung der Markierungen für die Tiefstwerte festgelegt. Die Länge ist dabei in Prozent bezogen auf die Rechteckbreite einzugeben. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(118 122 118 116 118 121 123; //Höchstwerte
               102 105 104 104 110 111 113; //Tiefstwerte
               104 119 118 108 115 118 113; //Schlusswerte
               113 121 114 112 118 121 121) //Eröffnungsw.
     CandlestickChart(;300;50;right;50;left)
     FillStyle(1;black)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(10 10 10 10 10; // Höchstwerte
                0  0  0  0  0; // Tiefstwerte
                5  3  8  3  1; // "Schlusswerte"
                2  9  7  5  1) // "Eröffnungswerte"
     CandlestickChart(horizontal;60;100;center;100;center)
     FillStyle(1;darkYellow)
     GridLocation(all;none)     // kein Raster
     AxisLine(x;0)              // keine x-Achsenlinie
     AxisMajorTicks(x;;;;;out)  // Skala verschieben
     AxisOptions(y;none)        // keine y-Achse
CloseDrawing()
```

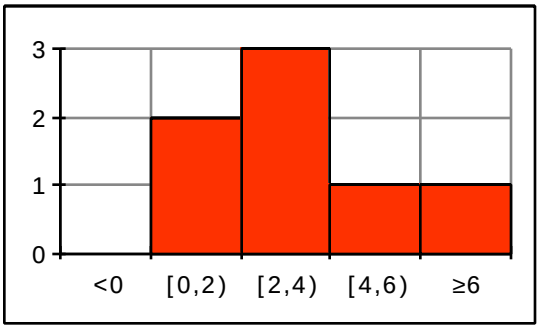


Histogramme:

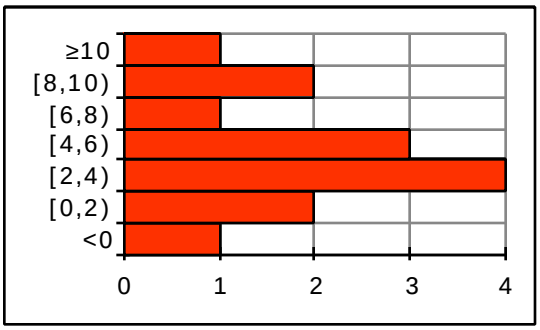
Histogram(darstellung;kategorienabstand;serienabstand)
HistogramRange(minWert;maxWert;intervallanzahl)

Die Funktion `Histogram()` dient zur Auswertung und Darstellung von Häufigkeitsverteilungen. Standardmäßig wird der Wertebereich durch den größten und kleinsten Wert in der Funktion `ChartData()` bestimmt und in zehn Intervalle unterteilt. Mit den Argumenten *minWert*, *maxWert* und *intervallanzahl* in der Funktion `HistogramRange()` können sowohl der auszuwertende Bereich als auch die Intervallanzahl vom Benutzer vorgegebenen werden. Dabei ist zu beachten, dass standardmäßig noch zwei Randintervalle hinzugefügt werden, d.h. zum Beispiel bei *intervall-anzahl=3* werden insgesamt fünf Intervalle dargestellt. Die Beschriftung der Intervalle erfolgt durch die in Klammern angeführten Intervallgrenzen. Zum Beispiel, "[2,4)" bedeutet, das Intervall umfaßt alle Werte größer gleich zwei und kleiner vier. Zu beachten ist, dass die Funktion `Histogram-Range()` nach der Funktion `Histogram()` anzuführen ist. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(6.5 0 2.9 1.5 3 2.5 5.9)
     Histogram()
     HistogramRange(0;6;3) // nach Histogram()!
CloseDrawing()
```

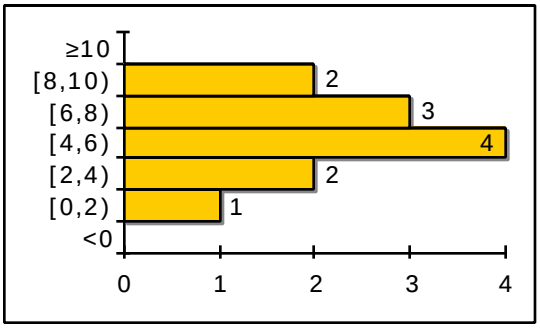



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 2 0 3 8 4 5 -2 6 9 5 2 12 2)
     Histogram(horizontal)
     HistogramRange(0;10;5) // nach Histogram()!
     ScalingOptions(x;;on)  // nur ganzahlige x-Skala
CloseDrawing()
```

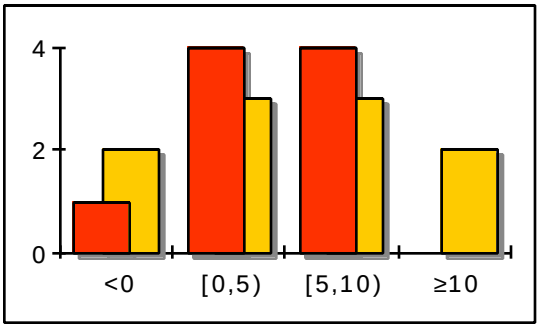


Das Aussehen der Balken wird durch die Argumente *darstellung*, *kategorienabstand* und *serienabstand* in der Funktion `Histogram()` kontrolliert. Diese Argumente werden ausführlich bei den *Balkendiagrammen* behandelt. Beispiele:

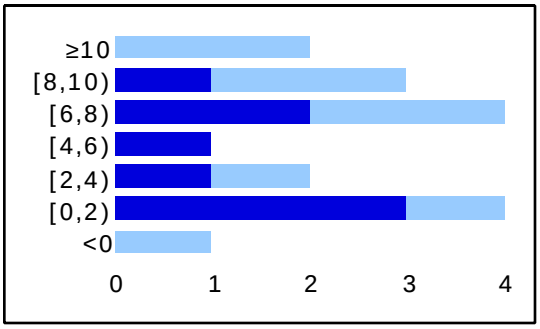
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(2.1 4.8 0 7 9.9 5 7.2 6 4 8 2.5 5.1)
     Histogram(horizontal+shadow+label)
     HistogramRange(0;10;5) // nach Histogram()!
     FillStyle(1;darkYellow)
     ShadowStyle(1;1;gray)
     ScalingOptions(x;;on) // nur ganzahlige x-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9.9 5 7.2 3 0.9 6 -.5; // 1.Serie
               6 9 2 7 2 12 -2 0.5 11 -1) // 2.Serie
     Histogram(shadow;50;-50)
     HistogramRange(0;10;2) // nach Histogram()!
     FillStyle(2;darkYellow)
     ShadowStyle(all;2;gray)
     ScalingOptions(y;;on) // nur ganzahlige y-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```



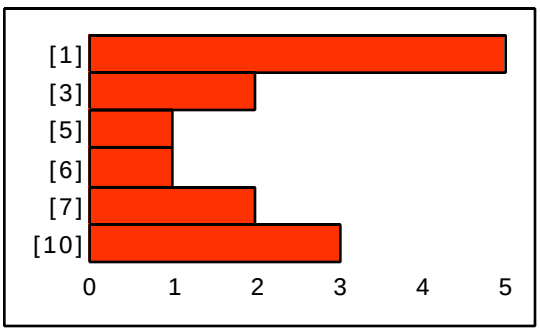

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9.9 5 7.2 3 0.9 6;  // 1.Serie
               6 9 8 7 2 12 -2 0.5 11) // 2.Serie
     Histogram(horizontal+stacked;40)
     HistogramRange(0;10;5) // nach Histogram()!
     FillStyle(1;blue)
     FillStyle(2;lightBlue)
     BorderStyle(all;none)
     AxisLine(all;0)       // keine Achsenlinien
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     ScalingOptions(x;;on) // nur ganzahlige x-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```



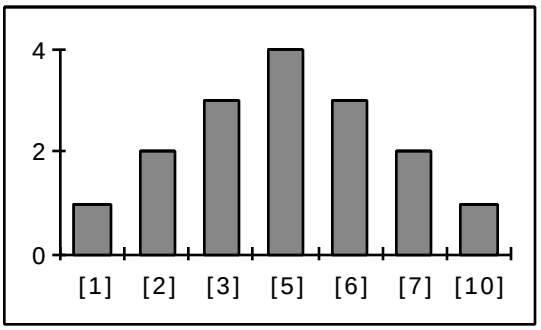
HistogramOptions(werteZählen;inHöheresIntervall;endenHinzufügen;frequenzlinien)

Optional können die Werte auch ohne Intervallbereiche, das heißt allein nach ihrer Häufigkeit, ausgewertet werden. Dies wird durch Aktivieren des ersten Arguments *werteZählen=on* ermöglicht. Die Funktion `HistogramRange()` wird in diesem Fall ignoriert, genauso wie die Argumente *inHöheresIntervall* und *endenHinzufügen*.
Die Funktion `HistogramOptions()` ist genauso wie die Funktion `HistogramRange()` nach der Funktion `Histogram()` anzuführen. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 10 7 10 1 5 3 7 3 1 1 6 10 1)
     Histogram(horizontal)
     HistogramOptions(on)
     AxisLine(all;0)       // keine Achsenlinien
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     ScalingOptions(y;on)  // y-Skala von oben nach unten
     ScalingOptions(x;;on) // nur ganzahlige x-Skala
     GridLocation(xy;none)
CloseDrawing()
```

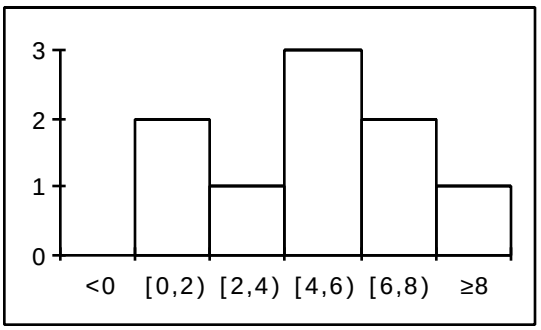


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(6 2 5 7 3 6 5 3 7 3 2 5 1 6 10 5)
     Histogram(;80)
     HistogramOptions(on) // nach Histogram()!
     FillStyle(1;gray)
     GridLocation(xy;none)
CloseDrawing()
```

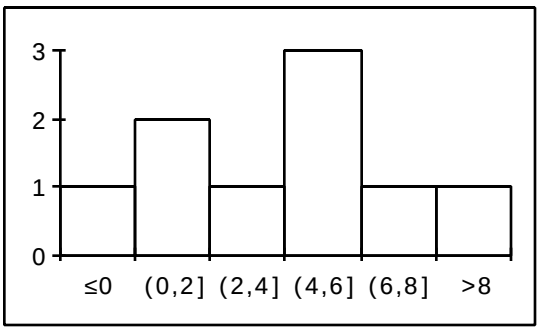


Mit dem zweiten Argument *inHöheresIntervall* kann die Zuordnung jener Werte, welche genau an einer Intervallgrenze liegen, kontrolliert werden. Standardmäßig werden diese "Grenzwerte" dem höheren Intervall zugeordnet (*inHöheresIntervall=on*), bei *inHöhersIntervall=off* dem niedrigeren Intervall. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9 5 7 4 2 6 5)
     Histogram()
     HistogramRange(0;8;4) // nach Histogram()!
     HistogramOptions(;on) // nach Histogram()!
     FillStyle(1;;transparent)
     ScalingOptions(y;;on) // nur ganzahlige y-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```

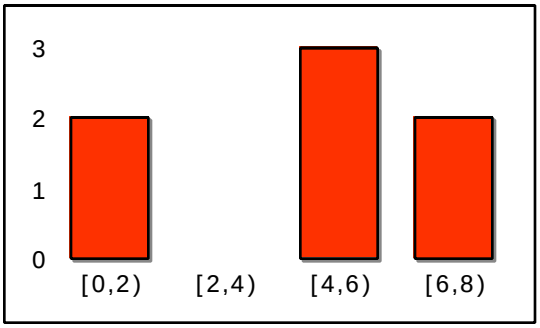


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9 5 7 4 2 6 5)
     Histogram()
     HistogramRange(0;8;4)  // nach Histogram()!
     HistogramOptions(;off) // nach Histogram()!
     FillStyle(1;;transparent)
     ScalingOptions(y;;on) // nur ganzahlige y-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```

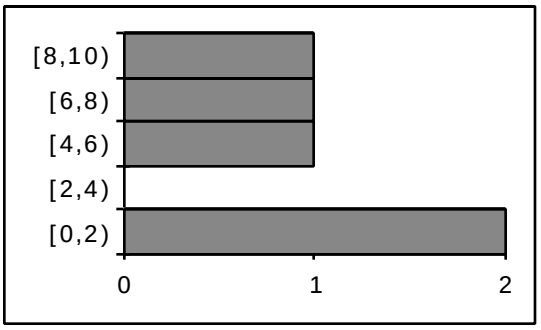


Die standardmäßig hinzugefügten Randintervalle können mit dem 3. Argument *endenHinzufügen=off* unterdrückt werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9 5 7 4 8 6 5)
     Histogram(shadow;50)
     HistogramRange(0;8;4)   // nach Histogram()!
     HistogramOptions(;;off) // nach Histogram()!
     ShadowStyle(1;1;gray)
     AxisLine(all;0)       // keine Achsenlinien
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     ScalingOptions(x;;on) // nur ganzahlige y-Skala
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 0 9.9 5 7.2)
     Histogram(horizontal)
     HistogramOptions(;;off) // nach Histogram()!
     HistogramRange(0;10;5)  // nach Histogram()!
     FillStyle(all;gray)
     GridLocation(xy;none)
     ScalingOptions(x;;on) // nur ganzahlige x-Skala
CloseDrawing()
```



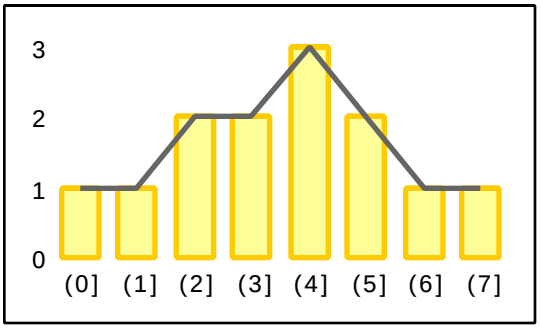
Optional können Histogramme durch Frequenzlinien ergänzt werden. Dazu stehen als 4. Argument *frequenzlinien* die folgenden Konstanten zur Verfügung:

Konstante	*Wert*	*Anmerkung*
`none`	`0`	keine Frequenzlinien
`frequency`	`1`	Frequenzlinien
`ogive`	`2`	laufend aufsummierte Frequenzen
`reverseOgive`	`3`	rückwärts laufende Frequenzsummen

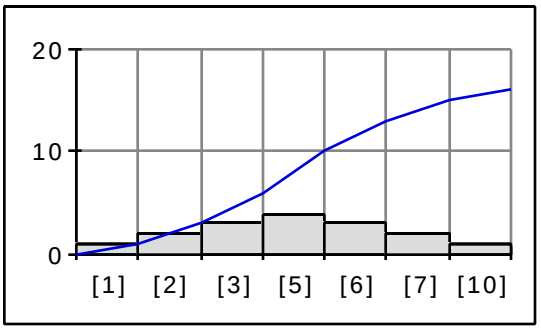
Die Frequenzlinien können durch die Funktion `LineStyle()` gestaltet werden. Die Funktion `LineStyle()` wird im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4 9.9 5 4.2 3 4.9 6
               6 9 8 7 2 12 3 3.5 11)
     Histogram(;50)
     HistogramRange(0;8;8) // nach Histogram()!
     HistogramOptions(;off;off;frequency)
     FillStyle(1;lightYellow)
     BorderStyle(1;;2;darkYellow)
     LineStyle(1;poly;2;darkGray)
     GridLocation(xy;none) // kein Raster
     AxisLine(all;0)       // keine Achsenlinien
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     AxisMajorTickLabelTexts(x;"(|u|]")
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(6 2 5 7 3 6 5 3 7 3 2 5 1 6 10 5)
     Histogram()
     HistogramOptions(on;;;ogive) // nach Histogram()
     FillStyle(1;lightGray)
     LineStyle(1;poly;1;blue)
CloseDrawing()
```



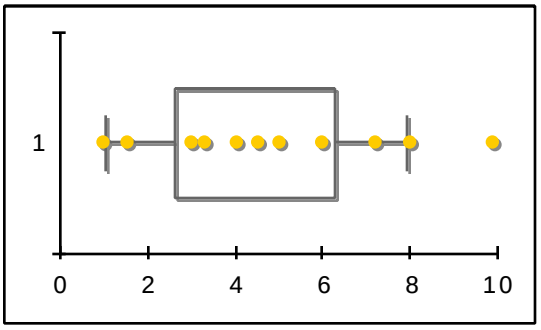
Box Plots:

Box Plots, auch als Box- und Whisker Plots bezeichnet, werden zur Darstellung statistischer Verteilungen verwendet.

**BoxPlots(darstellung;obereBoxPerzentile;
untereBoxPerzentile;obereWhiskerPerzentile;
untereWhiskerPerzentile)**

Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*) und Schatten (*darstellung=shadow*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Die Darstellung der Berandung, Symbole und Schatten kann durch die Stilfunktionen `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden. Alle Stilfunktionen werden im Abschnitt *Stile* behandelt. Beispiel:

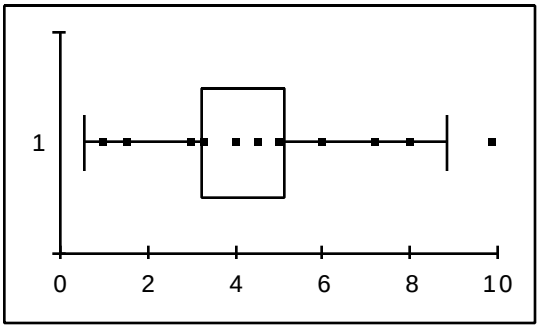
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1.5 8 9.9 4.5 7.2 3.3 4 5 3 6)
     BoxPlot(symbol+shadow+horizontal)
     SymbolStyle(all;bullet;4;1;darkYellow)
     BorderStyle(all;;1;darkGray)
     ShadowStyle(all;1;gray)
     GridLocation(xy;none)
CloseDrawing()
```



Die Box-Grenzwerte werden durch die Argumente *obereBoxPerzentile* (default: 75 Prozent) und *untereBoxPerzentile* (default: 25 Prozent) festgelegt, die Whisker-Grenzwerte werden durch die Argumente *obereWhiskerPerzentile* (default: 90 Prozent) und *untereWhiskerPerzentile* (default: 10 Prozent).

Beispiel:

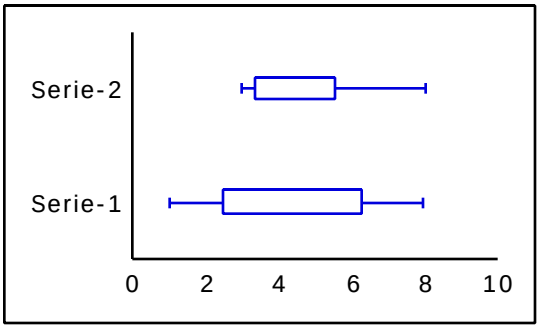
```
OpenDrawing(200;120)
   AddFrame(0;0;200;120)
   ChartData(1 1.5 8 9.9 4.5 7.2 3.3 4 5 3 6)
   BoxPlot(symbol+horizontal;65;35;95;5)
   SymbolStyle(all;square;2;1;black)
   GridLocation(xy;none)
CloseDrawing()
```



**BoxPlotOptions(boxabstand;istPerzentileGraph;boxFüllen;
mittelwertZeigen;medianZeigen;nurAusreißerZeigen;
nurMarkerZeigen;markerlänge)**

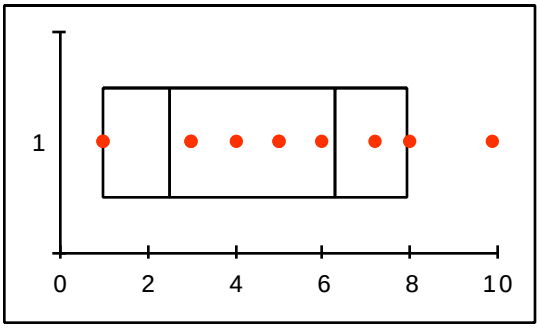
Die Funktion `BoxPlotOptions()` ist nach der Funktion `BoxPlot()` anzuführen und eröffnet eine Vielzahl von Gestaltungsmöglichkeiten. Mit dem 1. Argument *boxabstand* kann der Abstand zwischen den einzelnen Serien kontrolliert werden. Der Abstand wird dabei in Prozent der Rechteckbreite angegeben. Standardmäßig ist der Abstand gleich der Rechteckbreite (*boxabstand=100*). Bei Werten größer 100 (z.B. *boxabstand=400*) werden die Rechtecke schmäler und die Abstände dazwischen größer, bei einem Wert kleiner 100 (z.B. *boxabstand=50*) werden die Rechtecke breiter und die Abstände kleiner. Beispiel:

```
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6;     // 1.Serie
              3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
    BoxPlot(horizontal)
    BoxPlotOptions(400) // nach BoxPlot()!
    BorderStyle(all;;1;blue)
    AxisMajorTicks(all;0)
    AxisMajorTickLabelTexts(y;"Serie-1";"Serie-2")
    GridLocation(xy;none)
CloseDrawing()
```



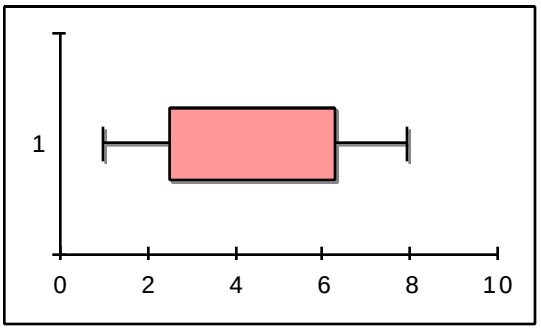
Durch Aktivierung des 2. Arguments *istPerzentileGraph=on* wird die Rechteckbox bis zur Whisker-Perzentile durchgezogen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
     BoxPlot(horizontal+symbol)
     BoxPlotOptions(;on) // nach BoxPlot()!
     SymbolStyle(1;bullet;4)
     GridLocation(xy;none)
CloseDrawing()
```



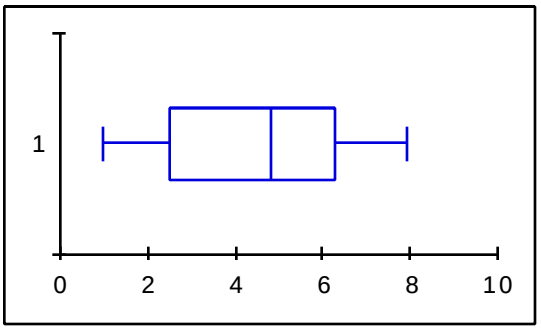
Mit dem 3. Argument *boxFüllen=on* und der Funktion `FillStyle()` bzw. `PictureStyle()` kann die Füllung der Rechteckbox kontrolliert werden. Die Funktionen `FillStyle()` und `PictureStyle()` werden im Abschnitt *Stile* erläutert. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
     BoxPlot(horizontal+shadow)
     BoxPlotOptions(200;off;on) // nach BoxPlot()!
     FillStyle(1;lightRed)
     BorderStyle(1;;1)
     ShadowStyle(1;1;gray)
     GridLocation(xy;none)
CloseDrawing()
```

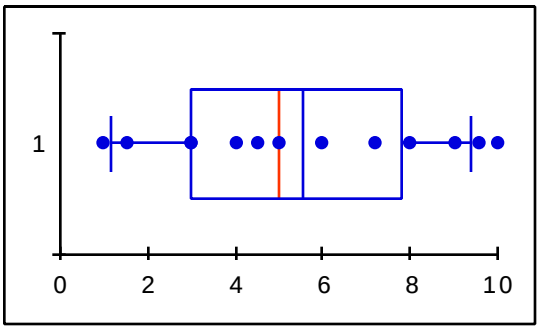


Weiters können Markierungslinien für den arithmetischen Mittelwert bzw. Median durch Aktivieren des Arguments *mittelwertZeigen=on* bzw. *medianZeigen=on* hinzugefügt werden. Die Darstellung der Mittelwertlinie ist ident der Berandung und wird durch die Funktion `BorderStyle()` kontrolliert. Das Aussehen der Medianlinie kann durch die Funktion `LineStyle()` kontrolliert werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
     BoxPlot(horizontal)
     BoxPlotOptions(200;off;off;on) // nach BoxPlot()
     BorderStyle(1;;1;blue)
     GridLocation(xy;none)
CloseDrawing()
```

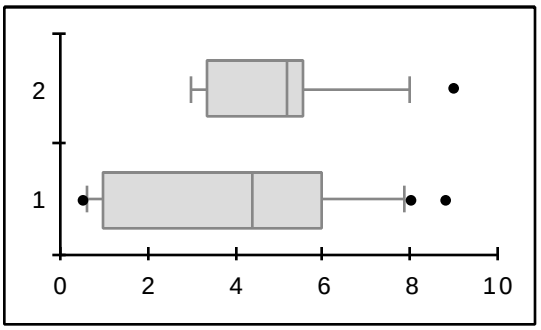



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 1 8 9.6 5 7.2 10 9 3 4 4.5 3 6)
     BoxPlot(horizontal+symbol)
     BoxPlotOptions(100;off;off;on;on) // nach BoxPlot()
     BorderStyle(1;;1;blue)
     LineStyle(1;;1;red)
     SymbolStyle(1;bullet;4;1;blue)
     GridLocation(xy;none)
CloseDrawing()
```



Durch Aktivieren des Arguments *nurAusreißerZeigen=on* kann die Darstellung der Symbole auf die "Ausreißer" beschränkt werden, d.h. auf jene Werte, welche außerhalb der Whisker-Perzentile liegen. Die Darstellung der Symbole wird durch *darstellung=symbol* in der Funktion `BoxPlot()` aktiviert. Beispiel:

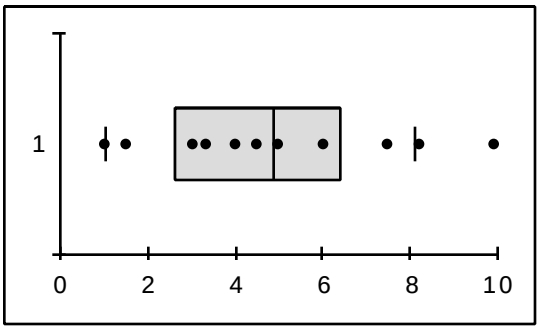
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(0.5 1 1 8 8.8 5 7.2 3 4 5 3 6; // 1.Serie
               3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
     BoxPlot(horizontal+symbol)
     BoxPlotOptions(100;off;on;on;off;on) //nach BoxPlot()
     FillStyle(all;lightGray)
     BorderStyle(all;;1;gray)
     SymbolStyle(all;bullet;3;;black)
     GridLocation(xy;none)
CloseDrawing()
```



Mit dem Argument *nurMarkerZeigen=on* kann die Verbindungslinie zwischen Rechteckbox und Whisker-Markierung ausgeblendet werden.

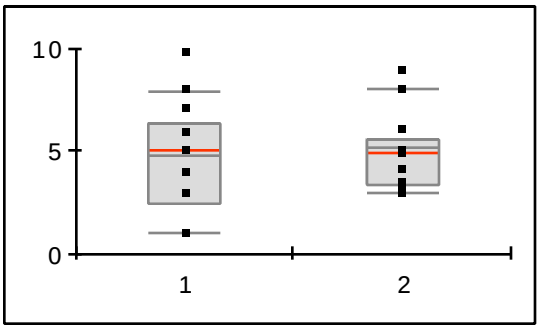
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1.5 8.2 9.9 5 7.5 3 4 4.5 3.3 6)
     BoxPlot(horizontal+symbol)
     BoxPlotOptions(200;off;on;on;off;off;on)
     FillStyle(1;lightGray)
     BorderStyle(1;;1;)
     SymbolStyle(1;bullet;3;;black)
     GridLocation(xy;none)
CloseDrawing()
```



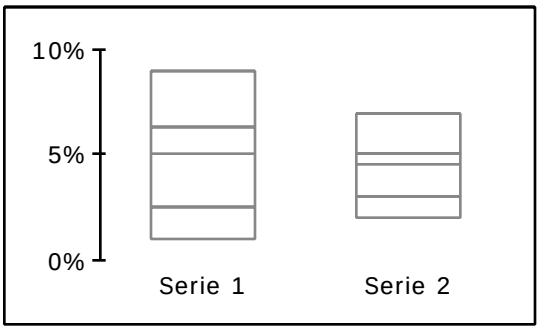
Die Markerlänge der Whisker-Perzentile kann mit dem letzten Argument *markerLänge* kontrolliert werden. Die Markerlänge ist in Prozent der Rechteckbreite anzugeben und ist standardmäßig 50, d.h. die halbe Rechteckbreite. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6;     // 1.Serie
               3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
     BoxPlot(symbol)
     BoxPlotOptions(200;off;on;on;on;off;on;100)
     FillStyle(all;lightGray)
     BorderStyle(all;;1;gray)
     LineStyle(all;;1;red) // Median
     SymbolStyle(all;square;2;;black)
     GridLocation(xy;none)
CloseDrawing()
```



Box Plots können auch zur Darstellung von Quartilen verwendet werden. Dazu wird die obere Box-Perzentile gleich 75% (das entspricht der 3. Quartile) und die untere Box-Perzentile gleich 25% gesetzt (das entspricht der 1. Quartile). Die 2. Quartile (=50% Perzentile) entspricht dem Median. Die Quartilen werden begrenzt durch die obere Whisker-Perzentile gleich 100% und durch die untere Whisker-Perzentile gleich 0%. Beispiel:

```
OpenDrawing(200;120)
   AddFrame(0;0;200;120)
   ChartData(1 1 8 9 5 7.2 3 4 5 3 6;       // 1. Serie
             3 5 3 4.1 7 3 5 2 5 3.5 4.9 6) // 2. Serie
   BoxPlot(;75;25;100;0)
   BoxPlotOptions(;on;  // istPerzentileGraph
                  off; // keine Füllung
                  off; // kein Mittelwert
                  on)  // Median (= 2.Quartile)
   BorderStyle(all;;1;gray)
   LineStyle(all;;1;gray) // Median
   AxisLine(x;0)
   AxisMajorTicks(x;0)
   AxisMajorTickLabelTexts(y;"|u|%")
   GridLocation(xy;none)
   AxisMajorTickLabelTexts(x;"Serie |u|")
CloseDrawing()
```



Stile

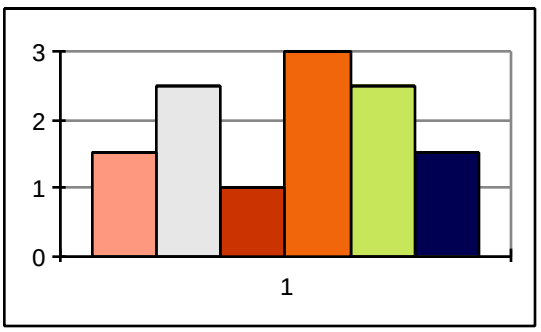
Stile erlauben die detaillierte Kontrolle über das Aussehen einzelner Diagrammserien. Folgende Komponenten können für jede Datenserie individuell gestaltet werden:

- Füllflächen
- Berandungen
- Linien
- Symbole
- Schatten
- Beschriftungen
- Pfeile

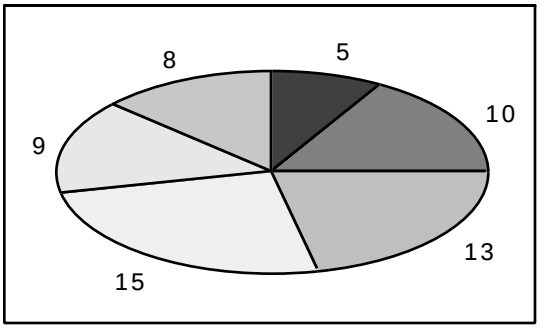
FillStyle(serienindex;farbe;muster)

Mittels der Funktion `FillStyle()` kann jeder Datenserie sowohl eine individuelle Füllfarbe als auch ein Füllmuster zugeordnet werden. Serien, denen nicht explizit ein Stil zugeordnet wird, werden in einer der 16 Standardfüllungen dargestellt. Ein Überblick über die vordefinierten Standardfüllungen sind in *xmReferenz* zu finden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5; 2.5; 1; 3; 2.5; 1.5)
     BarChart()
     FillStyle(1;red;gray)
     FillStyle(2;black;48)
     FillStyle(3;darkRed)
     FillStyle(4;;127)
     FillStyle(5;200 231 89)
CloseDrawing()
```

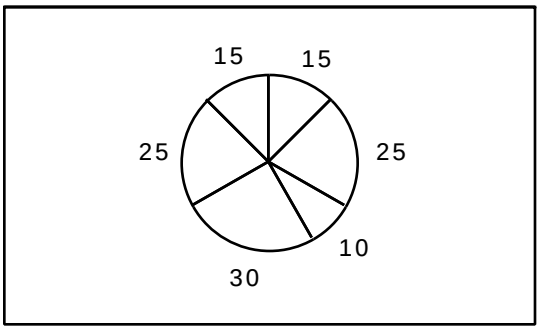


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 10 13 15 9 8)
     PieChart(oval+label)
     BorderStyle(all;)
     FillStyle(1;black;darkGray)
     FillStyle(2;black;gray)
     FillStyle(3;black;lightGray)
     FillStyle(4;black;13)
     FillStyle(5;black;48)
     FillStyle(6;black;53)
CloseDrawing()
```



Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, so wird allen Serien die gleiche Füllung zugewiesen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 10 30 25 15)
     PieChart(label)
     FillStyle(;;transparent)
CloseDrawing()
```



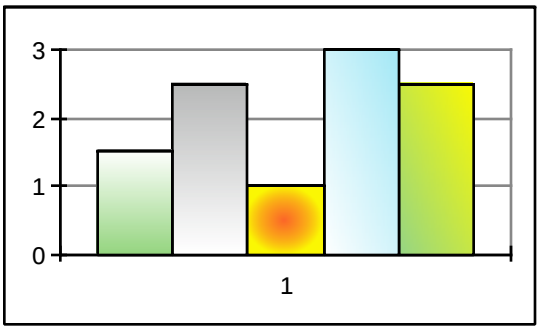
PictureStyle(serienindex;quelle;name; stapelnUndSkalieren)

Anstelle einer Füllfarbe kann wahlweise auch ein Bild als Füllung verwendet werden. Dabei werden drei verschiedene Bildquellen unterstützt:

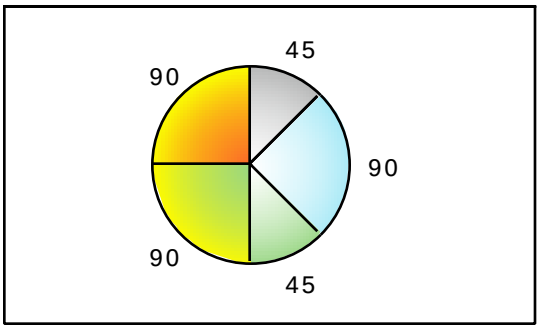
- *Zwischenablage:* (*quelle=clipboard*)
  Die Zwischenablage wird verwendet, wenn als Quelle die Konstante *clipboard* eingegeben wird. Das Argument *name* wird ignoriert.
  Beispiel: `PictureStyle(1;clipboard)`
- *Datei:* (*quelle=file*)
  Beispiel für MacOS/X:
  `PictureStyle(2;file;"Macintosh HD:Bilder:Bild1")`
  Unter MacOS/X können folgende Bildformate importiert werden: PICT, GIF, JPEG, PNG, BMP, TIFF.
  Beispiel für Windows:
  `PictureStyle(all;file;"C:\\Bilder\\Bild-1.bmp")`
  Unter Windows können folgende Bildformate importiert werden: WMF, EMF, GIF, JPEG, PNG, BMP, TIFF.
  Zu beachten ist, dass Sonderzeichen wie das Dateipfadtrenn-zeichen "\" durch einen vorangestellten Backslash "\\" eingegeben werden müssen. (Siehe *xmReferenz.pdf*, Abschnitt *Diverses*).
- *Resource*: (*quelle=resource*)
  Zur Zeit stehen insgesamt 42 in Resourcen abgespeicherte Bilder zur Verfügung. Der Zugriff erfolgt mittels einer Index-nummer zwischen 1 und 42. Dabei ist zu beachten, dass die Indexnummer unter Hochkomma zu setzen ist. Ein Überblick über die vordefinierten Bildresourcen findet sich in *xmReferenz*.
  Weiters ist unter Windows bei der Verwendung von Resourcen zu beachten, dass Diagramme zu einer Größe von mehreren MB anwachsen können. Dieses Problem tritt nur in Verbindung mit dem Standard-Ausgabeformat EMF auf, nicht jedoch bei der Ausgabe als Bitmap. EMF- und Bitmapformat werden im Zuge der Funktion `OpenDrawing()` im Abschnitt *Layout* behandelt.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5; 2.5; 1; 3; 2.5)
     BarChart()
     PictureStyle(1;resource;"5")
     PictureStyle(2;resource;"2")
     PictureStyle(3;resource;"42")
     PictureStyle(4;resource;"24")
     PictureStyle(5;resource;"30")
CloseDrawing()
```

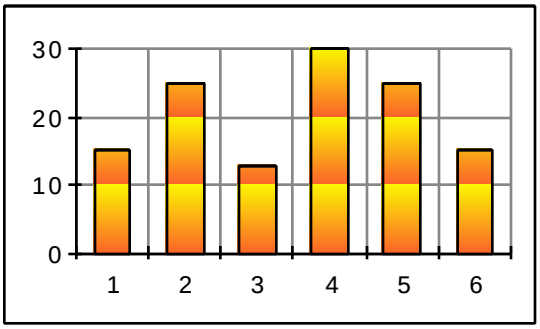



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 90 45 90 90)
     PieChart(label)
     PictureStyle(1;resource;"34")
     PictureStyle(2;resource;"36")
     PictureStyle(3;resource;"38")
     PictureStyle(4;resource;"40")
     PictureStyle(5;resource;"42")
CloseDrawing()
```



Bei Balkendiagrammen besteht zusätzlich noch die Möglichkeit, Bilder durch Vorgabe eines Skalierungswertes zu skalieren und zu stapeln. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 13 30 25 15)
     BarChart()
     PictureStyle(all;resource;"8";10)
CloseDrawing()
```



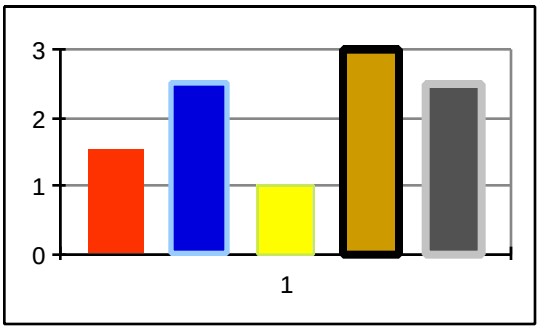
BorderStyle(serienindex;darstellung;strichstärke;farbe; muster)

Standardmäßig werden alle Ränder schwarz und ein Pixel breit dargestellt. Mit der Funktion `BorderStyle()` kann jeder Datenserie eine individuelle Berandung zugeordnet werden. Zusätzlich kann bei einigen Diagrammen über das Argument *darstellung* der Berandungsverlauf variiert werden.

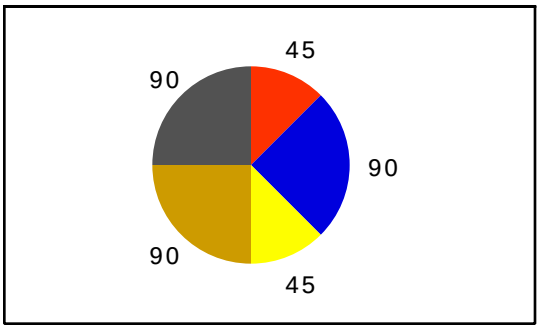
Konstante	*Wert*	*Diagrammart*
`none`	0	alle Diagramme
`jump`	1	*(wird nicht unterstützt)*
`step`	2	Areachart
`poly`	3	alle Diagramme *(default)*
`smooth`	4	Radarchart, Polarchart

Beispiele:

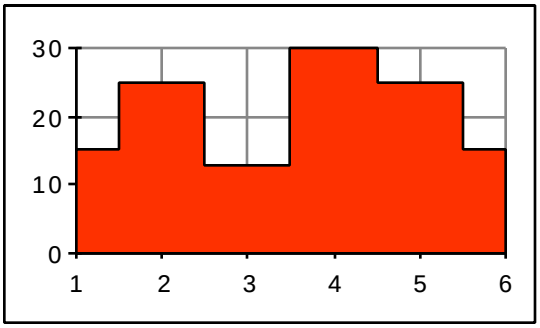
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5; 2.5; 1; 3; 2.5)
     BarChart(;;50)
     BorderStyle(1;poly;0)
     BorderStyle(2;poly;2;lightBlue)
     BorderStyle(3;poly;1;200 231 89)
     BorderStyle(4;;3)
     BorderStyle(5;;3;gray;gray)
CloseDrawing()
```



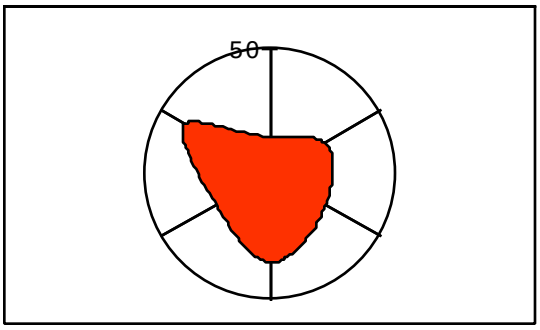
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(45 90 45 90 90)
     PieChart(label)
     BorderStyle(all;none)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 13 30 25 15)
     AreaChart()
     BorderStyle(1;step;1;black)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 25 35 25 40)
     RadarChart()
     MajorGridLinePatterns(all;all;black)
     BorderStyle(1;smooth)
CloseDrawing()
```



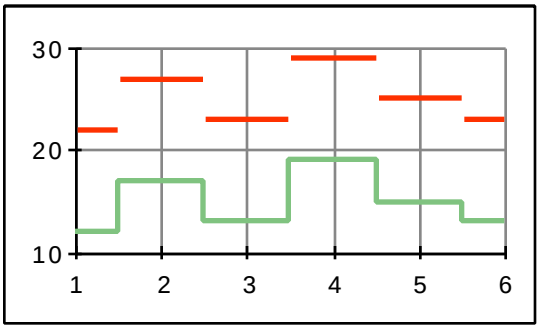
LineStyle(serienindex;darstellung;strichstärke;farbe; muster)

Mittels der Funktion `LineStyle()` kann jeder Datenserie ein individueller Linienstil zugeordnet werden. Serien, denen nicht explizit ein Stil zugeordnet wird, werden in einer der 16 Standardfarben dargestellt. Ein Überblick über die vordefinierten Farben sind in *xmReferenz* zu finden. Zusätzlich kann mit dem Argument *darstellung* der Kurvenverlauf variiert werden.

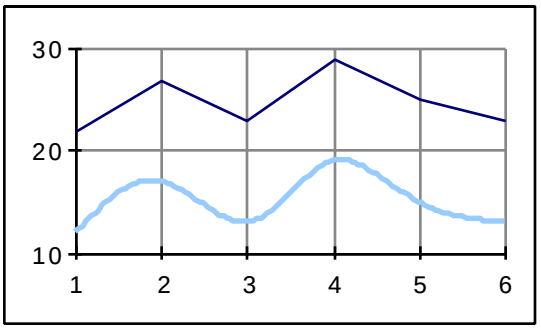
Konstante	*Wert*	*Darstellung*
none	0	Linienzug ausblenden
jump	1	nur horizontale oder vertikale Linien
step	2	stufenförmiger Linienzug
poly	3	geradliniger Linienzug *(default)*
smooth	4	geglätteter Linienzug

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(22 27 23 29 25 23;
               12 17 13 19 15 13)
     LineChart()
     LineStyle(1;jump;2)
     LineStyle(2;step;2;green;gray)
CloseDrawing()
```



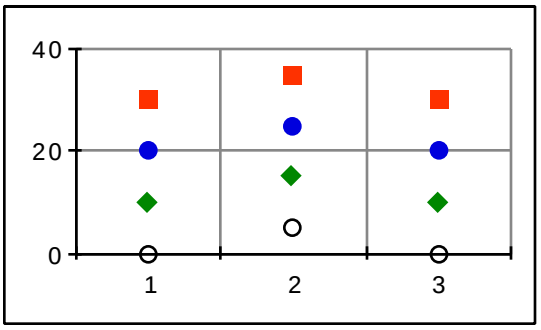
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(22 27 23 29 25 23;
               12 17 13 19 15 13)
     LineChart()
     LineStyle(1;poly;1;darkBlue)
     LineStyle(2;smooth;2;lightBlue)
CloseDrawing()
```



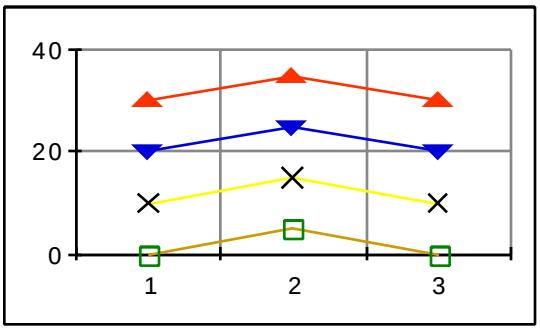
SymbolStyle(serienindex;typ;größe;strichstärke;farbe; muster)

Durch die Funktion `SymbolStyle()` kann jeder Datenserie ein individuelles Symbol zugeordnet werden. Serien, denen nicht explizit ein Symbol zugeordnet wird, werden in einem der 12 Standardsymbole dargestellt. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(30 35 30; 20 25 20; 10 15 10; 0 5 0)
     ScatterChart(;on)
     SymbolStyle(1;square;6)
     SymbolStyle(2;bullet;6)
     SymbolStyle(3;diamond;6;;green)
     SymbolStyle(4;circle;6;;black)
CloseDrawing()
```

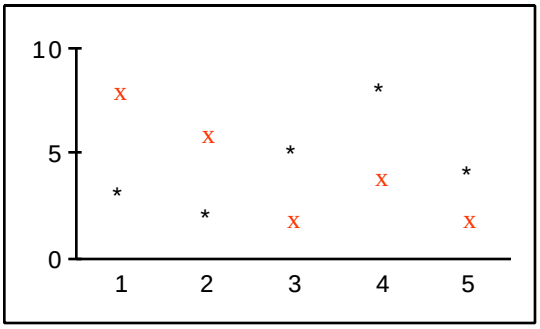


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(30 35 30;
               20 25 20;
               10 15 10;
                0  5  0)
     LineChart(symbol;on)
     SymbolStyle(1;upTriangle;10)
     SymbolStyle(2;downTriangle;10)
     SymbolStyle(3;cross;8;1;black)
     SymbolStyle(4;hollowSquare;6;1;green)
CloseDrawing()
```



Optional können als Symbole auch Zeichen aus einem Zeichensatz verwendet werden; dadurch ergibt sich eine Vielzahl neuer "Symbole", welche bei Verwendung von Postscript- oder TrueType-Fonts mit hoher Qualität ausgegeben werden können. Die Zeichen werden durch die Funktionen `LabelTexts()` und `LabelStyle()` festgelegt. Die Funktion `LabelOptions()` ermöglicht die exakte Platzierung der Zeichen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 5 8 4; 8 6 2 4 2; 6 9 8 2 7)
     ScatterChart(label;on)
     SymbolStyle(all;none)
     LabelOptions(all;centerCenter)
     LabelTexts(1;"*")
     LabelTexts(2;"x")
     LabelStyle(2;"Times";10;;red)
     LabelTexts(3;"\xD1") // \xnn ASCII Code (hexadezimal)
     LabelStyle(3;"Symbol";12;;blue)
     AxisMajorTicks(x;0)
     GridLocation(;none)
CloseDrawing()
```



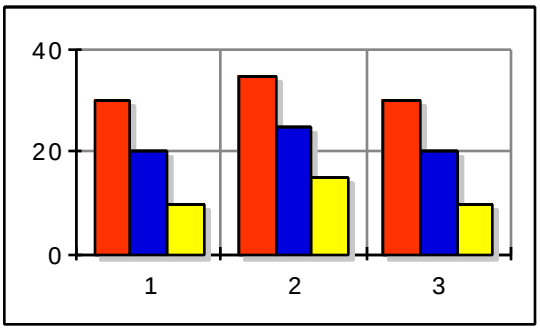
Symbole können auch unter Verwendung eines Blasendiagramms dargestellt werden. Dabei ergibt sich die Möglichkeit, mit der Funktion `PictureStyle()` Bilder aus einer Datei einzulesen und als Symbole zu verwenden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 1 6 8; 1 1 1 1;
               8 4 2 4; 1 1 1 1;
               6 9 8 2; 1 1 1 1)
     BubbleChart(;on)
     BubbleChartOptions(9)
     BorderStyle(all;0)
     AxisMajorTicks(x;0)
     GridLocation(;none)

     // Beispiel für MacOS/X:
     // Unterstützte Dateiformate:
     // PICT, GIF, JPEG, PNG, BMP, TIFF
     PictureStyle(1;file;"Symbol01.pct")
     PictureStyle(2;file;"Macintosh HD:Symbole:Symb1.gif")

     // Beispiel für Windows:
     // Unterstützte Dateiformate:
     // WMF, EMF, GIF, JPEG, PNG, BMP, TIFF
     PictureStyle(3;file;"C:\\Symbole\\Symbol03.png")
CloseDrawing()
```

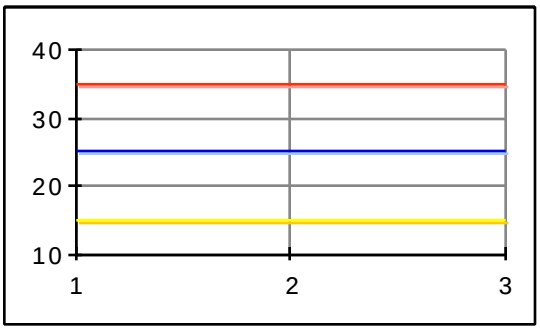
ShadowStyle(serienindex;abstand;farbe;muster)

Diagrammschatten werden standardmäßig in grau und in einem Abstand von drei Pixel dargestellt. Durch die Funktion `ShadowStyle()` kann die Darstellung des Schattens variiert werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(30 35 30;
               20 25 20;
               10 15 10)
     BarChart(shadow)
     ShadowStyle(all;2;200 200 200)
CloseDrawing()
```



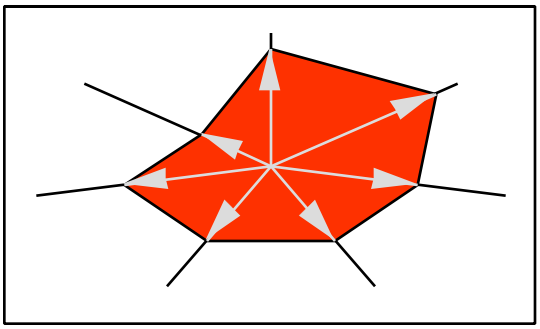

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(35 35 35; 25 25 25;15 15 15)
     LineChart(shadow)
     ShadowStyle(1;1;lightRed)
     ShadowStyle(2;1;lightBlue)
     ShadowStyle(3;1;darkYellow)
CloseDrawing()
```



**ArrowStyle(serienindex;strichstärke;farbe;muster;
pfeilposition;pfeillänge;pfeilbreite;pfeilkerbe;
istPfeilHohl)**

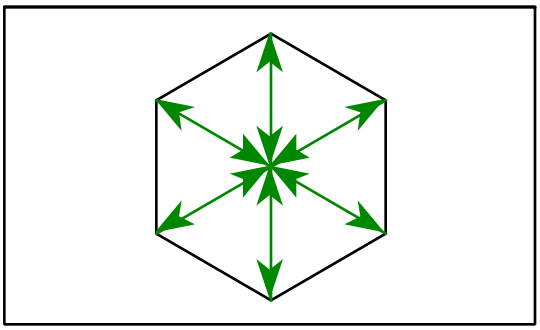
Radar- und Polardiagramme können optional durch Pfeile ergänzt werden. Diese verlaufen vom Zentrum zu den einzelnen Diagrammpunkten. Standardmäßig werden alle Pfeile in schwarz und 1 Pixel breit dargestellt. Durch die Funktion `ArrowStyle()` kann das Aussehen der Pfeile inkl. Pfeilspitzen für jede Datenserie getrennt kontrolliert werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(35 35 25 25 25 25 15)
     RadarChart(oval)
     // keine Skalenbeschriftung, kein Raster
     RadarChartOptions(0;none;on)
     ArrowStyle(1;1;lightGray)
CloseDrawing()
```

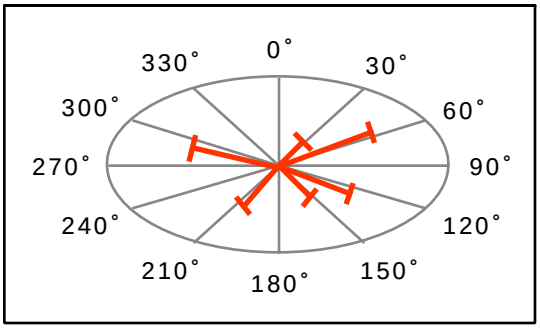


Weiters kann durch das Argument *pfeilposition* die Pfeilspitze am Ende (default), am Anfang oder auch beidseitig angeordnet werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(40 40 40 40 40 40)
     RadarChart()
     // keine Skalenbeschriftung, kein Raster
     RadarChartOptions(0;none;on)
     AxisLine(all;0)           // Achsen ausblenden
     FillStyle(1;;transparent) // Füllung ausblenden
     ArrowStyle(1;1;green;;begin+end;15;10;4)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(-10 21 7  -25 9 27; -22 -16 14 11 -18 19)
     PolarChart(oval)
     PolarChartOptions(0;;on) // keine Skalenbeschriftung
     ArrowStyle(1;2;red;;end;0)
     AxisLabelText(all;"|u|°")

     AxisLine(all;0)           // Achsen ausblenden
     BorderStyle(1;none)       // Rand ausblenden
     FillStyle(1;;transparent) // Füllung ausblenden
CloseDrawing()
```

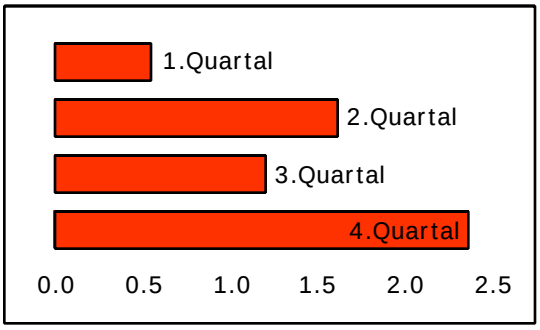


LabelTexts(serienindex;text1;text2...)

Die Funktion `LabelTexts()` ermöglicht die individuelle Beschriftung von Diagrammwerten. Die Texte können dabei sehr variable gestaltet werden:

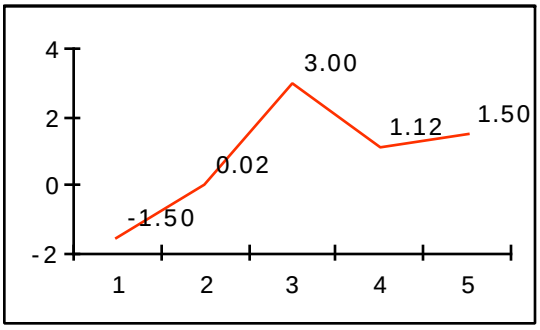
- Reiner Text:
  Zum Beispiel:

```
OpenDrawing(200;120)
      AddFrame(0;0;200;120)
      ChartData(0.5555 1.61 1.2 2.365)
      BarChart(label+horizontal;50)
      LabelTexts(1;"1.Quartal";
                  "2.Quartal";
                  "3.Quartal";
                  "4.Quartal")
      GridLocation(all;none)
      ScalingOptions(y;on)
      AxisOptions(y;none)
      AxisLine(x;0)
      AxisMajorTicks(all;0)
CloseDrawing()
```



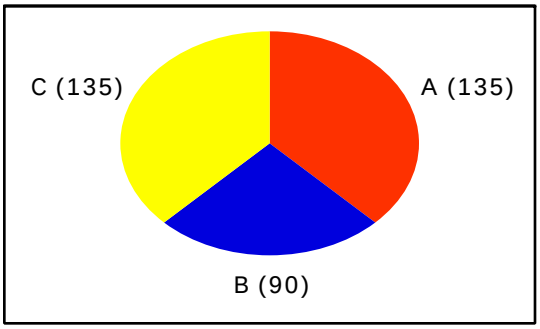
- Formatanweisung:
  Eine Formatanweisung ist stets zwischen zwei vertikale Balken "|" zu setzen. Eine detaillierte Erklärung aller Formatanweisungen inklusive zahlreicher Beispiele ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
      AddFrame(0;0;200;120)
      ChartData(-1.5 0.024 3 1.121 1.5)
      LineChart(label;on)
      LabelTexts(1;"|f2|")
      GridLocation(all;none)
CloseDrawing()
```



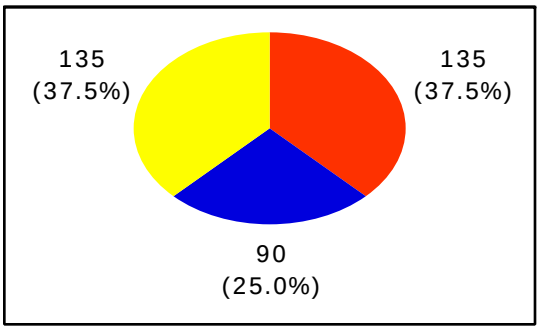
- Kombination aus Text und Formatanweisung:
  Vor und hinter einer Formatanweisung können optional noch Texte angefügt werden. Beispiel:

```
OpenDrawing(200;120)
      AddFrame(0;0;200;120)
      ChartData(135 90 135)
      PieChart(label+oval)
      BorderStyle(all;none)
      LabelTexts(1;"A (|u|)";"B (|u|)";"C (|u|)")
CloseDrawing()
```



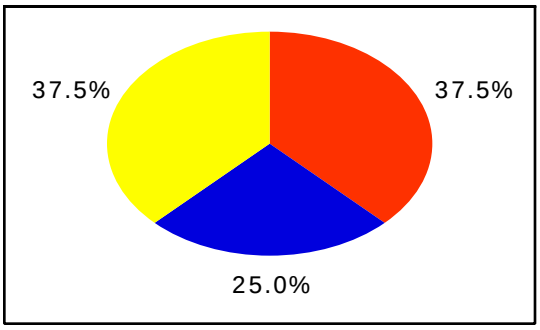
Bei Tortendiagrammen besteht zudem die Möglichkeit sowohl die absoluten Werte (default) als auch die prozentuellen Werte darzustellen. Dies geschieht durch Anführen von zwei Formatanweisungen. Die erste Formatanweisung beschreibt die absoluten Werte, die zweite Formatanweisung dient zur Formatierung der prozentuellen Werte. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(135 90 135)
     PieChart(label+oval)
     BorderStyle(all;none)
     LabelTexts(1;"|u|\n(|f1|%)")
     LabelStyle(all;;;;center)
CloseDrawing()
```



Sollen nur die prozentuellen Werte dargestellt werden, muß die Ausgabe der absoluten Werte durch eine leere Formatanweisung "||" unterdrückt werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(135 90 135)
     PieChart(label+oval)
     BorderStyle(all;none)
     LabelTexts(1;"|||f1|%")
CloseDrawing()
```

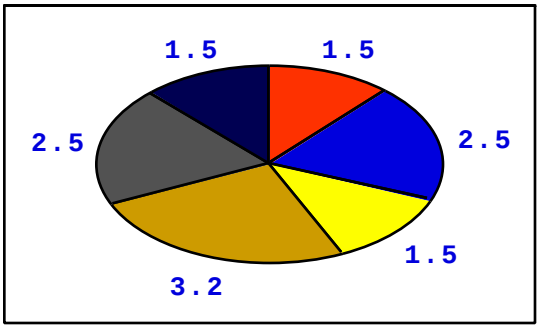


Die Texte bzw. Formatanweisungen werden periodisch wiederholt, falls die Anzahl der Diagrammwerte größer ist als die Anzahl der Textargumente. Wird kein Text übergeben, so wird die Default-Formatanweisung "|u|" verwendet. Weiters sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "\n" möglich.

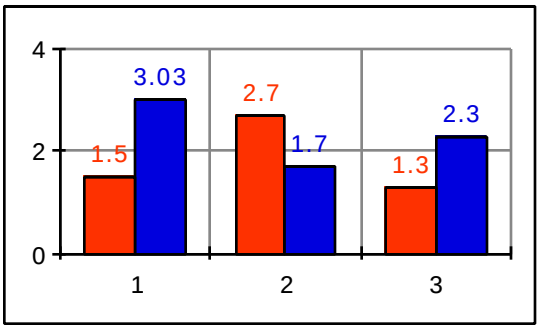
LabelStyle(serienindex;schrift;größe;stil;farbe; ausrichtung)

Mit der Funktion `LabelStyle()` kann den Diagrammwerten ein Schriftstil zugeordnet werden. Dabei kann für jede Datenserie ein individueller Schriftstil festgelegt werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.5 3.2 2.5 1.5)
     PieChart(oval+label;)
     LabelStyle(1;"Courier";10;bold;blue)
CloseDrawing()
```



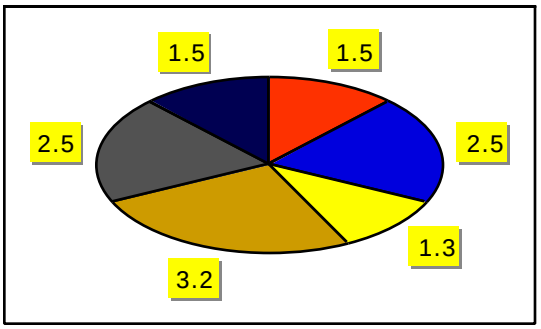

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.7 1.3; 3.03 1.7 2.3)
     BarChart(label)
     LabelStyle(1;;9;;red)
     LabelStyle(2;;9;;blue)
CloseDrawing()
```



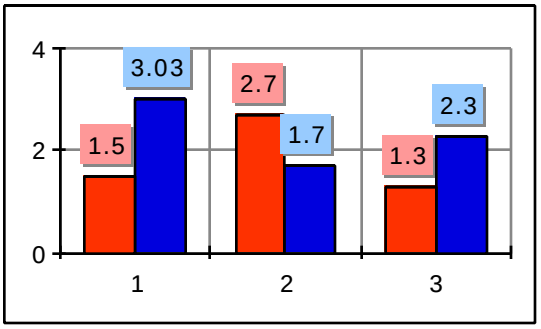
LabelBackground(serienindex;füllfarbe;füllmuster;
rahmenbreite;rahmenfarbe;rahmenmuster;
schattenabstand;schattenfarbe;schattenmuster)

Durch die Funktion `LabelBackground()` kann der Beschriftungshintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.3 3.2 2.5 1.5)
     PieChart(oval+label)
     LabelBackground(1;yellow;;0;;;1)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.7 1.3; 3.03 1.7 2.3)
     BarChart(label)
     LabelBackground(1;lightRed;;0;;;1)
     LabelBackground(2;lightBlue;;0;;;1)
CloseDrawing()
```



LabelOptions(serienindex;platzierung;hVersatz;vVersatz; unteresLimit;oberesLimit)

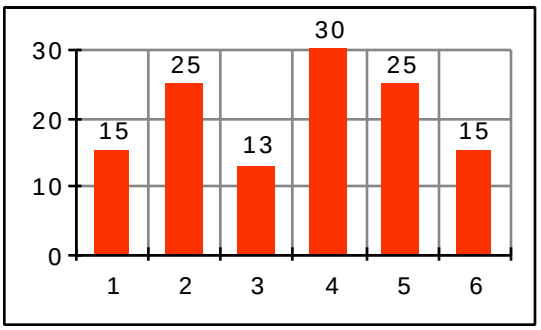
Die Funktion `LabelOptions()` stellt zahlreiche Möglichkeiten zur Platzierung und Feinabstimmung der Beschriftung zur Verfügung:

- *Platzierung der Beschriftung bei nicht gestapelten Balkendiagrammen:* Ist genügend Platz vorhanden, so wird standardmäßig die Beschriftung außerhalb der Balken platziert, ansonsten innerhalb. Mit dem Argument *platzierung* können bei nicht gestapelten Balkendiagrammen, nicht gestapelten Histogrammen und Gantt-Diagrammen neun Positionen definiert werden:

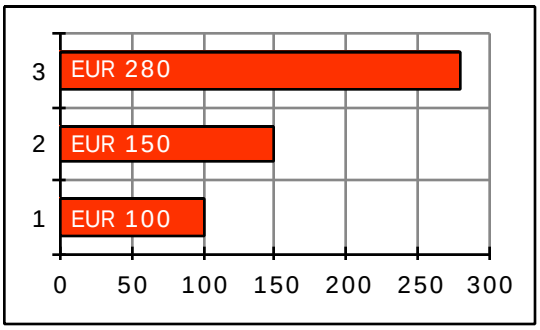
Konstante	*Wert*	*Platzierung*
`smartBegin`	1	Balkenanfang, falls genügend Platz
`smartCenter`	2	Balkenmitte, falls genügend Platz
`smartEnd`	3	Balkenende, falls genügend Platz
`begin`	4	Balkenanfang (zwingend)
`center`	5	Balkenmitte (zwingend)
`end`	6	Balkenende (zwingend)
`edge`	7	Rand am Balkenende (zwingend)
`smartOut`	8	außerhalb, falls genügend Platz (default)
`out`	9	außerhalb (zwingend)

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 13 30 25 15)
     BarChart(label)
     BorderStyle(all;0)
     LabelOptions(1;out)
CloseDrawing()
```



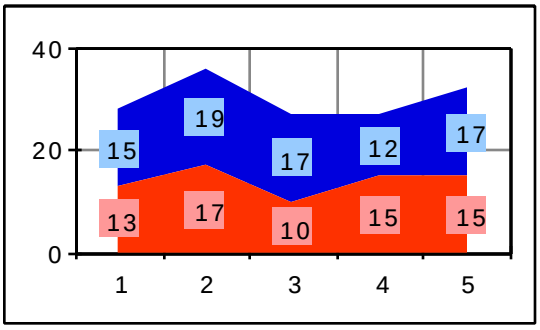
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(100 150 280)
     BarChart(label+horizontal)
     LabelTexts(1;"EUR |u|")
     LabelStyle(;;;bold;white)
     LabelOptions(1;begin)
CloseDrawing()
```



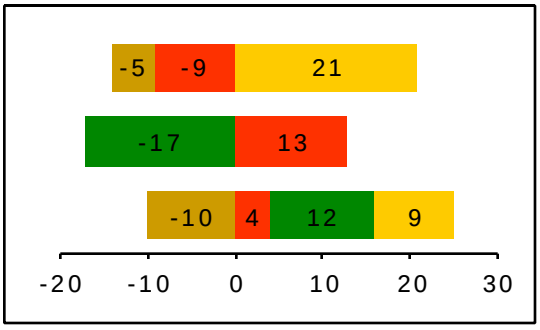
Weitere Beispiele finden sich im Abschnitt *Diagramme*.

- *Platzierung der Beschriftung bei gestapelten Diagrammen:*
  Bei gestapelten oder proportionalen Balken- oder Flächendiagrammen wird die Beschriftung standardmäßig in der Mitte der einzelnen Balken- bzw. Flächenabschnitte platziert; das Argument *platzierung* wird ignoriert. Beispiele:

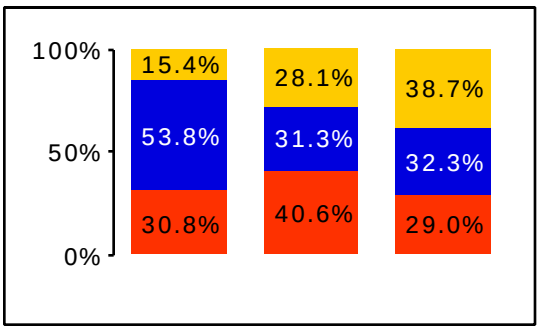
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(13 17 10 15 15; 15 19 17 12 17)
     AreaChart(stacked+label;on)
     BorderStyle(all;none)
     LabelBackground(1;lightRed;;0)
     LabelBackground(2;lightBlue;;0)
     GridFrame()
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4  13 -9;
               12 -17  0;
                9   0 21;
              -10   0 -5)
     BarChart(stacked+horizontal+label;50)
     BorderStyle(all;none)
     FillStyle(2;green)
     FillStyle(3;darkYellow)
     GridLocation(all;none) // kein Raster
     AxisOptions(y;none)    // keine y-Achse
     AxisMajorTicks(x;2;;;;out)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 13  9;
               7 10 10;
               2  9 12)
     BarChart(proportional+label;40)
     Scaling(y;percent)
     BorderStyle(all;none) // keine Balkenberandung
     FillStyle(3;darkYellow)
     LabelTexts(all;"|2f1|%")
     LabelStyle(2;;;;white)
     GridLocation(all;none) // kein Raster
     AxisOptions(x;none)    // keine x-Achse
     AxisMajorTicks(y;2;;;;out)
CloseDrawing()
```



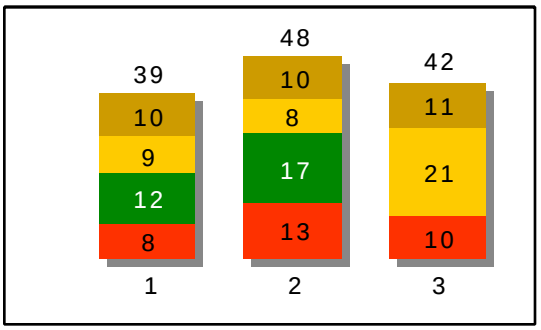
Zusätzlich besteht die Möglichkeit, entweder die Gesamtsummen oder die laufenden Summen darzustellen. Die Darstellung und Platzierung der (laufenden) Summen werden durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()`, `LabelOptions()` in Verbindung mit *serienindex=-1* bzw. *serienindex=stacked* kontrolliert. Dieser spezielle Serienindex steht ausschließlich in den vier oben angeführten Stilfunktionen in Verbindung mit gestapelten oder proportionalen Diagrammen zur Verfügung. Dabei können die (laufenden) Summen entweder außerhalb (default) oder am Rand der einzelnen Balken- bzw. Flächenabschnitte platziert werden. Dazu stehen für das Argument *platzierung* die folgenden vier Konstanten zur Verfügung:

Konstante	*Wert*	*Platzierung*
`totalsOut`	1	die Gesamtsummen außerhalb
`totalsEdge`	2	die Gesamtsummen am Rand
`runningTotalsOut`	3	die laufenden Summen außerhalb
`runningTotalsEdge`	4	die laufenden Summen am Rand

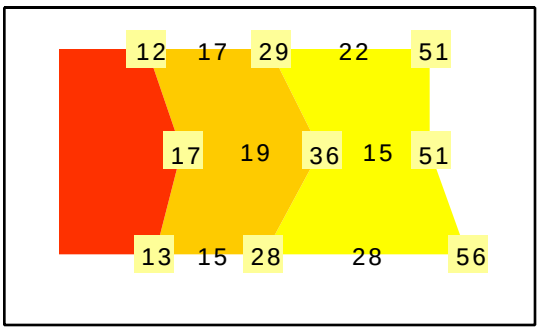
Die sonst üblichen Platzierungskonstanten werden bei *serienindex=-1* bzw. *serienindex=stacked* ignoriert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 8  13 10;
               12  17  0;
                9   8 21;
               10  10 11)
     BarChart(stacked+label+shadow;50)
     LabelOptions(stacked;totalsOut)
     BorderStyle(all;none)
     LabelStyle(2;;;;white)
     FillStyle(2;green)
     FillStyle(3;darkYellow)
     GridLocation(all;none) // kein Raster
     AxisOptions(y;none)    // keine y-Achse
     AxisMajorTicks(x;0)    // keine x-Achsenmarkierungen
     AxisLine(x;0)          // keine x-Achsenlinie
CloseDrawing()
```

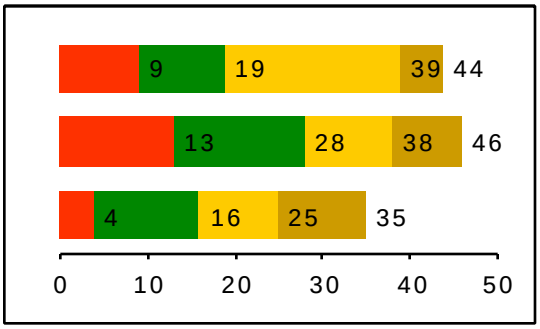



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(13 17 12;
               15 19 17;
               28 15 22)
     AreaChart(stacked+label+horizontal)
     LabelTexts(1;" ")
     BorderStyle(all;none)
     FillStyle(2;darkYellow)
     LabelOptions(-1;runningTotalsEdge)
     LabelBackground(stacked;lightYellow;;0)
     GridLocation(all;none) // Raster ausblenden
     AxisOptions(all;none)  // Achsen ausblenden
CloseDrawing()
```

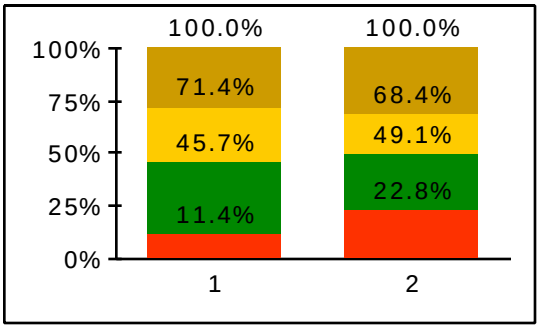


Falls nur die Endsummen bzw. die laufenden Summen dargestellt werden sollen, kann man die "standardmäßige" Beschriftung durch Vorgabe einer "leeren Beschriftung", z.B. `LabelTexts(all;" ")`, ausblenden. Beispiele:

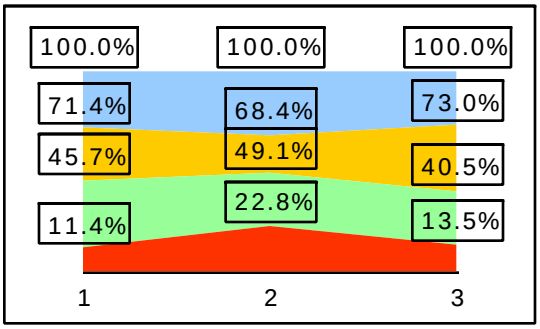
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9;
               12 15 10;
                9 10 20;
               10  8  5)
     BarChart(stacked+horizontal+label;50)
     LabelTexts(all;" ")
     LabelOptions(stacked;runningTotalsOut)
     BorderStyle(all;none)
     FillStyle(2;green)
     FillStyle(3;darkYellow)
     GridLocation(all;none) // kein Raster
     AxisOptions(y;none)    // y-Achse ausblenden
     AxisMajorTicks(x;2;;;out)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13;
               12 15;
                9 11;
               10 18)
     BarChart(proportional+label;50)
     Scaling(y;percent;0;100;4)
     LabelTexts(all;" ")
     LabelTexts(stacked;"|2f1|%")
     LabelOptions(stacked;runningTotalsOut)
     BorderStyle(all;none)
     FillStyle(2;green)
     FillStyle(3;darkYellow)
     GridLocation(all;none) // kein Raster
     AxisMajorTicks(x;0)
CloseDrawing()
```

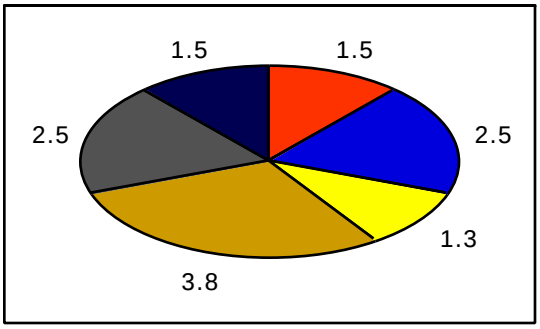



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenChart(30;25;140;75;on)
     ChartData(4 13 5; 12 15 10; 9 11 12; 10 18 10)
     AreaChart(proportional+label)
     LabelTexts(all;" ")
     LabelTexts(stacked;"|2f1|%")
     LabelBackground(stacked;;transparent)
     LabelOptions(stacked;runningTotalsOut)
     BorderStyle(all;none)
     FillStyle(2;lightGreen)
     FillStyle(3;darkYellow)
     FillStyle(4;lightBlue)
     GridLocation(all;none)
     AxisOptions(y;none) // y-Achse ausblenden
     AxisMajorTicks(x;0)
CloseDrawing()
```



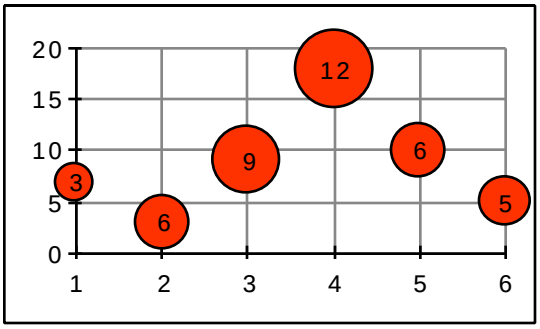
- *Platzierung der Beschriftung bei Tortendiagrammen:*
  Bei Tortendiagrammen werden die Werte stets außerhalb des Diagramms platziert — das Argument *platzierung* wird ignoriert. Es besteht aber die Möglichkeit mit den Funktionen `PieChartInnerLabelTexts()` und `PieChartCenterLabelText()` zusätzlich Texte auch innerhalb und im Zentrum von Tortendiagrammen zu platzieren. Detaillierte Informationen dazu inkl. Beispiele finden sich im Abschnitt *Diagramme*. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.3 3.8 2.5 1.5)
     PieChart(oval+label)
CloseDrawing()
```



- *Platzierung der Beschriftung bei Blasendiagrammen (bubble charts):* Standardmäßig wird die Beschriftung im Zentrum platziert (*platzierung=centerCenter*). Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 3 9 18 10 5; // Positionen
               3 6 9 12  6 5) // Durchmesser
     BubbleChart(label)
CloseDrawing()
```

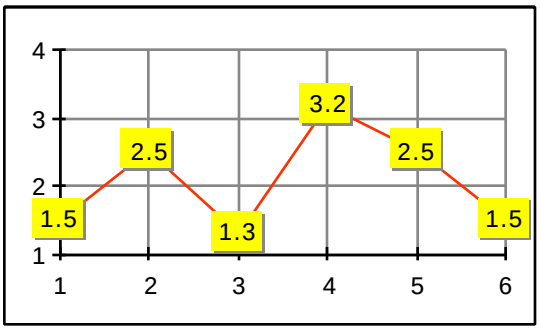


- *Platzierung der Beschriftung bei allen übrigen Diagrammen:* Standardmäßig wird die Beschriftung rechts oberhalb des Diagrammwerts positioniert. Mit dem Argument *platzierung* können neun Positionen definiert werden:

Konstante	*Wert*	*Platzierung*
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts (default)
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

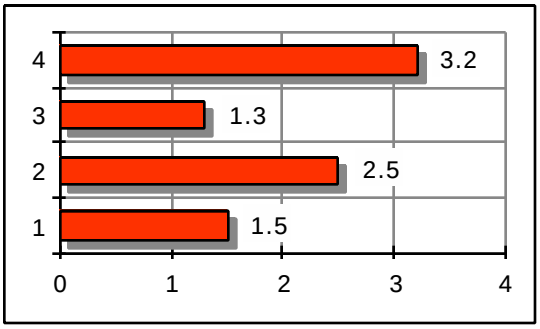
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.3 3.2 2.5 1.5)
     LineChart(label)
     LabelBackground(1;yellow;;0;;;1)
     LabelOptions(all;centerCenter)
CloseDrawing()
```



Optional kann noch mit den Argumenten *hVersatz* und *vVersatz* die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben. Beispiel:

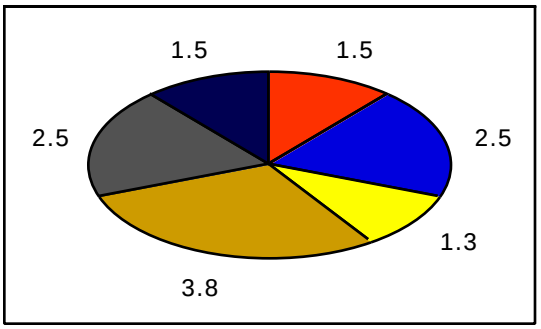
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.3 3.2)
     BarChart(label+horizontal+shadow)
     LabelBackground(1;white;;0)
     LabelOptions(1;;3) // 3 Pixel nach rechts
CloseDrawing()
```



Bei Tortendiagrammen werden die Argumente *hVersatz* und *vVersatz* ignoriert; stattdessen ist es möglich, mit der Funktion `PieChartLabelOptions(;textversatzAußen)` einen radialen Versatzwert zu definieren. Ein positiven Versatzwert vergrößert den Abstand zwischen Berandung und Beschriftung, ein negativer Wert verkleinert den Abstand. Das Argument *textversatzAußen* ist in Prozent der Segmentlänge einzugeben.

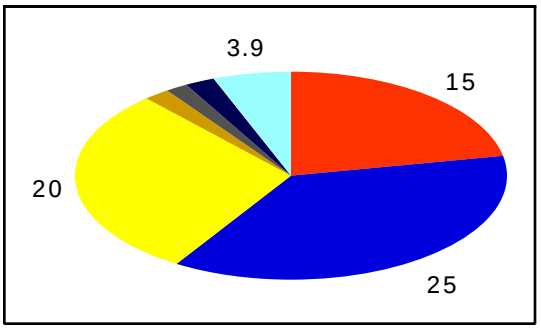
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.3 3.8 2.5 1.5)
     PieChart(oval+label)
     PieChartLabelOptions(;10)
CloseDrawing()
```



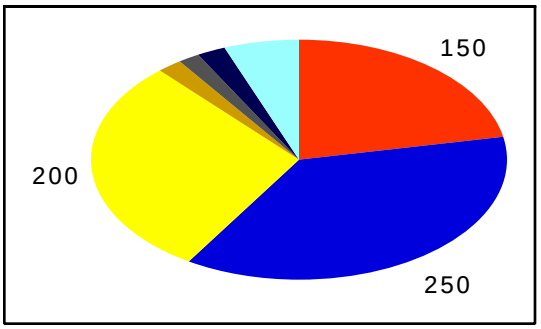
Mithilfe der Argumente *unteresLimit* und *oberesLimit* kann die Beschriftung von Diagrammwerten unterdrückt werden, d.h. nur Diagrammwerte größer als das untere Limit und kleiner als das obere Limit werden beschriftet. So kann zum Beispiel bei Tortendiagrammen durch Angabe eines unteren Limits die Beschriftung schmaler Segmente verhindert werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 25 20 1.3 1.2 1.5 3.9)
     PieChart(label+oval)
     BorderStyle(all;none)
     LabelOptions(1;;0;0;3.0)
CloseDrawing()
```



Zusätzlich besteht bei Tortendiagrammen die Möglichkeit, mittels der Funktion `PieChartLabelOptions(relativeLimitsVerwenden)` zwischen absoluten und relativen Grenzwerten zu wählen. Relative Grenzwerte sind in Prozent bezogen auf die Gesamtsumme anzugeben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(150 250 200 13 12 15 39)
     PieChart(label+oval)
     BorderStyle(all;none)
     // alle Werte, die kleiner sind als
     // 10% der Gesamtsumme ausblenden
     LabelOptions(1;;0;0;10.0)
     PieChartLabelOptions(on)
CloseDrawing()
```



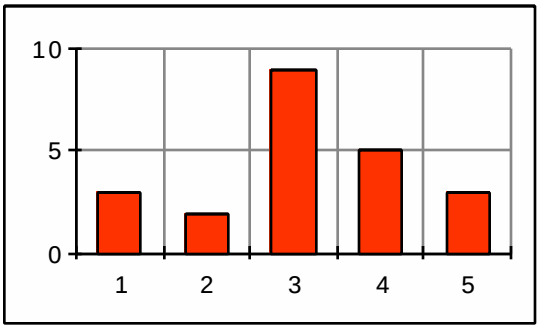
Hintergrund

Zur Gestaltung des Hintergrundes stehen insgesamt vier Funktionen zur Verfügung, je zwei zur Gestaltung des "Gesamthintergrundes", d.h. der Hintergrund erstreckt sich über die gesamte Zeichnung oder des umschließenden Views, und zwei weitere Funktionen zur Gestaltung des eigentlichen Diagramm-Hintergrundes, d.h. der Hintergrund ist auf den Bereich des Diagrammrasters beschränkt.

Background(füllfarbe;füllmuster;rahmenbreite;
rahmenfarbe;rahmenmuster;schattenabstand;
schattenfarbe;schattenmuster)

Die Funktion `Background()` erlaubt die Festlegung eines einfarbigen Hintergrundes inklusive Berandung und Schatten. Wird die Funktion `Background()` ohne Argumente angeführt, so wird ein weißer Hintergrund mit einem 1 Pixel breiten schwarzen Rand ohne Schatten gezeichnet. Beispiel:

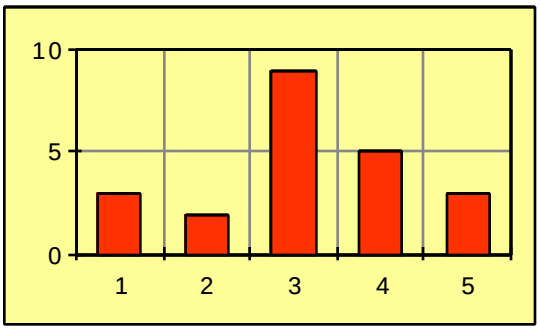
```
OpenDrawing(200;120)
     ChartData(3 2 9 5 3)
     BarChart()
     Background()
CloseDrawing()
```



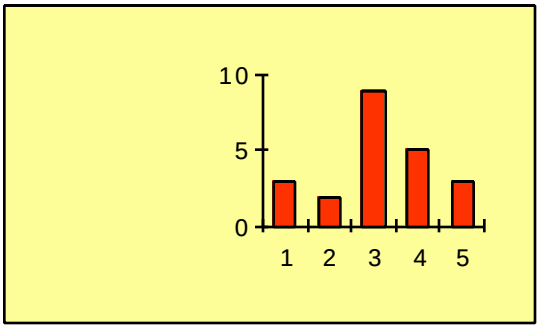
Werden keine Views verwendet oder wird die Hintergrundfunktion außerhalb der Views angeführt, bezieht sich die Funktion auf die gesamte Zeichnung. Views werden im Abschnitt *Layout* behandelt.

Beispiele:

```
OpenDrawing(200;120)
     ChartData(3 2 9 5 3)
     BarChart()
     GridFrame()
     Background(lightYellow)
CloseDrawing()
```



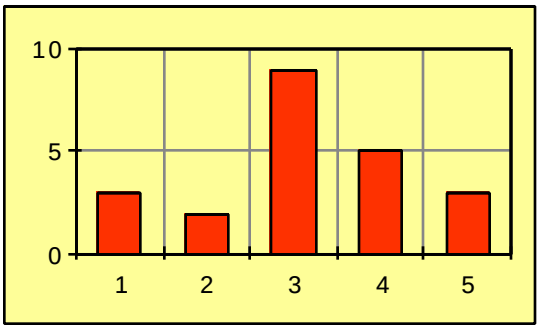

```
OpenDrawing(200;120)
     OpenView(70;10;120;100)
          ChartData(3 2 9 5 3)
          BarChart()
          GridLocation(all;none) // kein Raster
     CloseView()
     Background(lightYellow)
CloseDrawing()
```



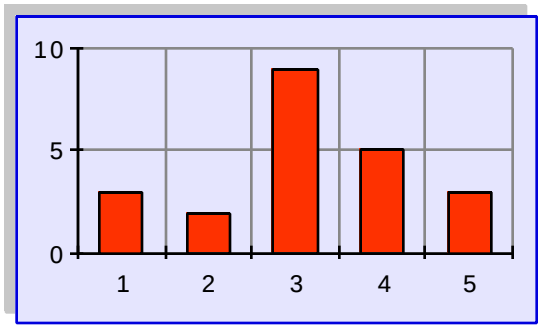
Ist der Schattenabstand positiv, so wird der Schatten rechts unten dargestellt, bei einem negativen Abstand, wird der Schatten links oben gezeichnet.

Beispiele:

```
OpenDrawing(200;120)
     ChartData(3 2 9 5 3)
     BarChart()
     Background(255 255 180;;0;;;4;lightGray)
CloseDrawing()
```

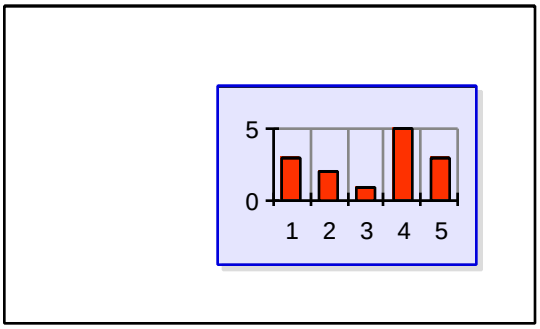



```
OpenDrawing(200;120)
     ChartData(3 2 9 5 3)
     BarChart()
     Background(230 230 255;;1;blue;;-4;200 200 200)
CloseDrawing()
```

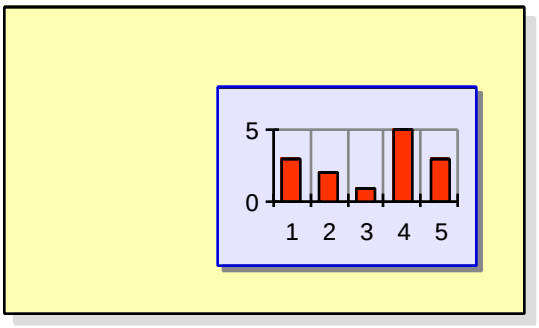


Wird die Funktion `Background()` innerhalb eines Views angeführt, so bezieht sich die Funktion auf den Hintergrund des Views. Beispiele:

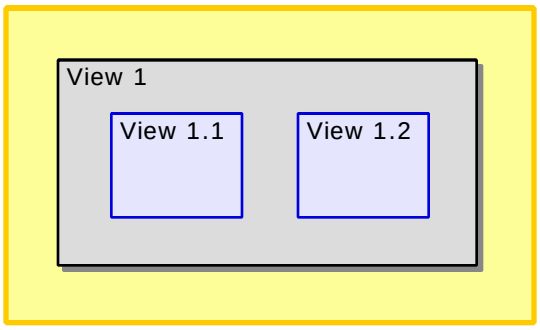
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenView(80;30;100;70)
          AddFrame(0;0;100;70)
          ChartData(3 2 1 5 3)
          BarChart()
          Background(230 230 255;;1;blue;;2;lightGray)
     CloseView()
CloseDrawing()
```




```
OpenDrawing(200;120)
     OpenView(80;30;100;70)
          AddFrame(0;0;100;70)
          ChartData(3 2 1 5 3)
          BarChart()
          Background(230 230 255;;1;blue;;2)
     CloseView()
     Background(255 255 180;;1;;;4;lightGray)
CloseDrawing()
```




```
OpenDrawing(200;120)
     OpenView(20;20;160;80)
          Background(lightGray;;1;;;2)
          AddText(3;10;"View 1")
          OpenView(20;20;50;40)
               Background(230 230 255;;1;blue)
               AddText(3;10;"View 1.1")
          CloseView()
          OpenView(90;20;50;40)
               Background(230 230 255;;1;blue)
               AddText(3;10;"View 1.2")
          CloseView()
     CloseView()
     Background(lightYellow;;2;darkYellow)
CloseDrawing()
```



BackgroundPict(quelle;name;platzierung;angleichen; istProportional)

Die Funktion `BackgroundPict()` ermöglicht die Verwendung eines Bildes als Hintergrund. Dabei werden drei verschiedene Bildquellen unterstützt:

- *Zwischenablage*: (*quelle=clipboard*)
  Die Zwischenablage wird verwendet, wenn als Quelle die Konstante *clipboard* eingegeben wird oder das argument *quelle* leer bleibt. Das 2. Argument *name* hat keine Bedeutung und wird ignoriert.
  Beispiele: `BackgroundPict(clipboard)`
  `BackgroundPict()`
- *Datei*: (*quelle=file*)
  Beispiel für MacOS/X:
  `BackgroundPict(file;"Macintosh HD:Bilder:Bild-1")`
  Unter MacOS/X können folgende Bildformate importiert werden: PICT, GIF, JPEG, PNG, BMP, TIFF
  Beispiel für Windows:
  `BackgroundPict(file;"C:\\Bilder\\Bild-1.jpg")`
  Unter Windows können folgende Bildformate importiert werden: WMF, EMF, GIF, JPEG, PNG, BMP, TIFF. Zu beachten ist, dass Sonderzeichen wie das Dateipfad-Trennzeichen "\" durch einen vorangestellten Backslash "\\" eingegeben werden müssen.
- *Resource*: (*quelle=resource*)
  Zur Zeit stehen insgesamt 42 in Resourcen abgespeicherte Farbverlaufbilder zur Verfügung. Der Zugriff erfolgt mittels einer Indexnummer zwischen 1 und 42. Dabei ist zu beachten, dass die Indexnummer unter Hochkomma zu setzen ist. Ein Überblick über die vordefinierten Bildresourcen findet sich in *xmReferenz*.
  Weiters ist unter Windows bei der Verwendung von Resourcen zu beachten, dass Diagramme zu einer Größe von mehreren MB anwachsen können. Dieses Problem tritt nur in Verbindung mit dem Standard-Ausgabeformat EMF auf, nicht jedoch bei der Ausgabe als Bitmap. EMF- und Bitmapformat werden im Zuge der Funktion `OpenDrawing()` im Abschnitt *Layout* behandelt.

Das dritte Argument *platzierung* ermöglicht neun verschiedene Positionen zur Platzierung eines Hintergrundbildes.

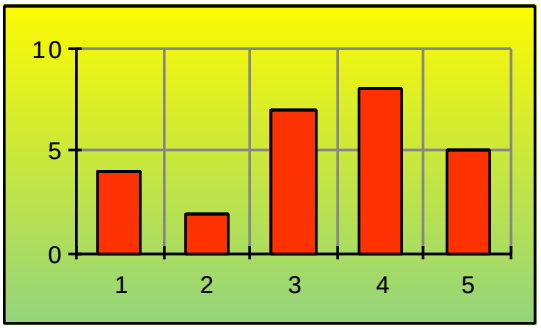
Konstante	*Wert*	*Anmerkung*
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte (default)
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Das 4. Argument *angleichen* stellt fünf Optionen zum Angleichen des Bildes an die Hintergrundfläche zur Verfügung.

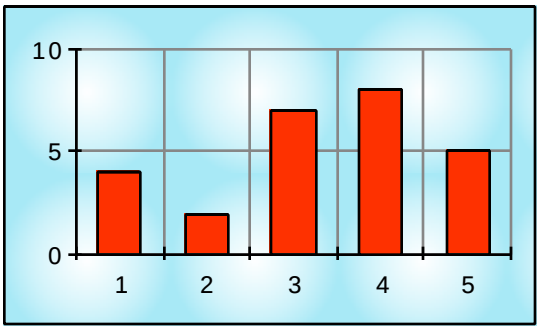
Konstante	*Wert*	*Anmerkung*
crop	1	abschneiden
reduce	2	verkleinern
enlarge	3	vergrößern
reduceOrEnlarge	4	verkleinern oder vergrößern (default)
tile	5	aneinander reihen

Zusätzlich kann mit dem 5. Argument *istProportional* festgelegt werden, ob das Verhältnis Bildbreite zu Bildhöhe beim Anpassen an die Hintergrundfläche beibehalten werden soll. Bei *angleichen=crop* und *angleichen=tile* wird das Argument *istProportional* ignoriert. Beispiele:

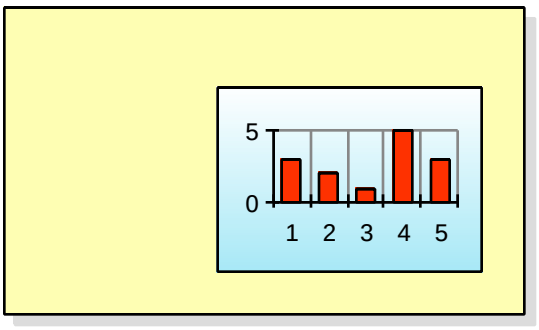
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     BackgroundPict(resource;"7")
CloseDrawing()
```



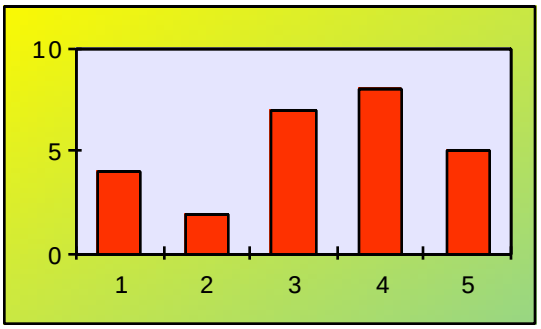
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     BackgroundPict(resource;"36";;tile)
CloseDrawing()
```



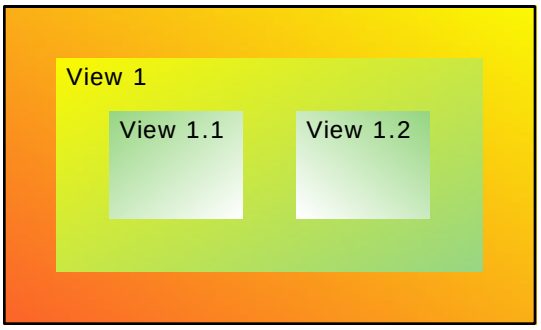

```
OpenDrawing(200;120)
     OpenView(80;30;100;70)
          AddFrame(0;0;100;70)
          ChartData(3 2 1 5 3)
          BarChart()
          BackgroundPict(resource;"3")
     CloseView()
     Background(255 255 180;;1;;;4;lightGray)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     GridLocation(all;none) // kein Raster
     ChartBackground(all;230 230 255)
     BackgroundPict(resource;"29")
CloseDrawing()
```

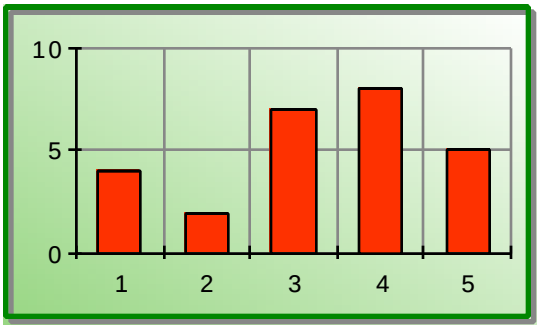



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenView(20;20;160;80)
          BackgroundPict(resource;"29")
          AddText(3;10;"View 1")
          OpenView(20;20;50;40)
               BackgroundPict(resource;"26")
               AddText(3;10;"View 1.1")
          CloseView()
          OpenView(90;20;50;40)
               BackgroundPict(resource;"28")
               AddText(3;10;"View 1.2")
          CloseView()
     CloseView()
     BackgroundPict(resource;"32")
CloseDrawing()
```



Weiters besteht die Möglichkeit, Hintergrundfunktionen zu kombinieren. Beispiel:

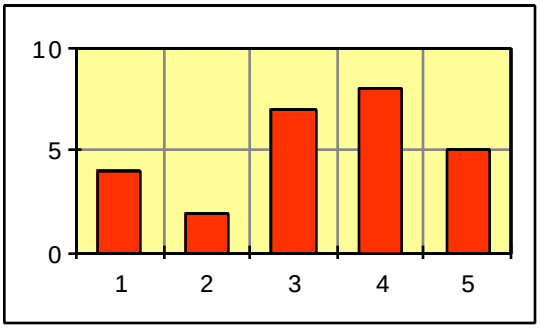
```
OpenDrawing(200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     BackgroundPict(resource;"27")
     Background(;transparent;2;green;;2)
CloseDrawing()
```



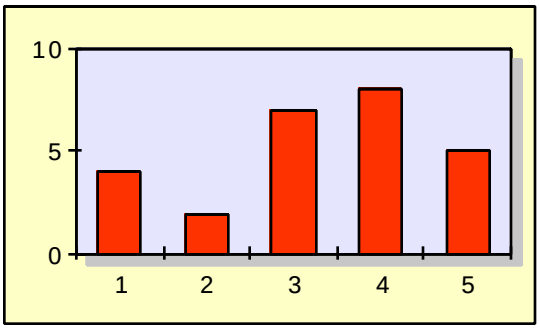
ChartBackground(ebenenindex;füllfarbe;füllmuster;
rahmenbreite;rahmenfarbe;rahmenmuster;
schattenbreite;schattenfarbe;schattenmuster)

Die Funktion `ChartBackground()` ist bis auf das 1. Argument *ebenenindex* ident aufgebaut wie die Funktion `Background()`. Im Gegensatz zur Funktion `Background()` ist die Hintergrundfläche bei der Funktion `ChartBackground()` auf den Bereich des Diagrammrasters begrenzt. Als erstes Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Ansonst gelten die gleichen Überlegungen wie bei der Funktion `Background()` beschrieben. Beispiele:

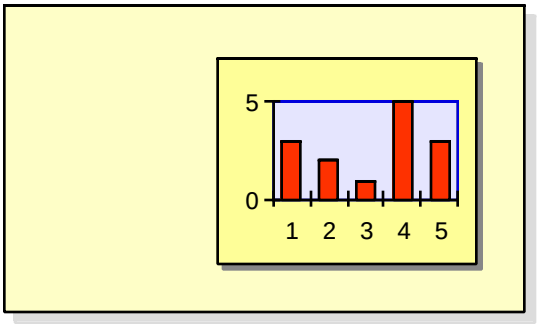
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     GridFrame()
     ChartBackground(xy;lightYellow)
CloseDrawing()
```



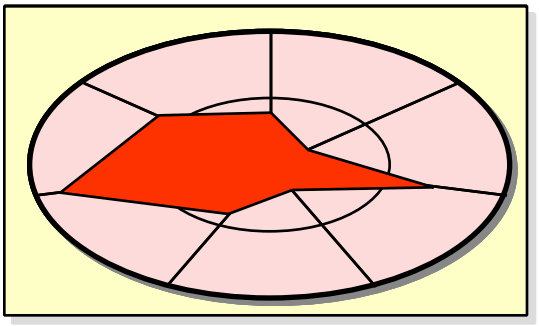

```
OpenDrawing(200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     GridLocation(all;none) // kein Raster
     ChartBackground(;230 230 255;;1;black;;4;200 200 200)
     Background(255 255 200)
CloseDrawing()
```




```
OpenDrawing(200;120)
     OpenView(80;20;100;80)
          ChartData(3 2 1 5 3)
          BarChart()
          GridLocation(all;none) // kein Raster
          ChartBackground(xy;230 230 255;;1;blue)
          Background(lightYellow;;;;;2)
     CloseView()
     Background(255 255 200;;1;;;4;lightGray)
CloseDrawing()
```



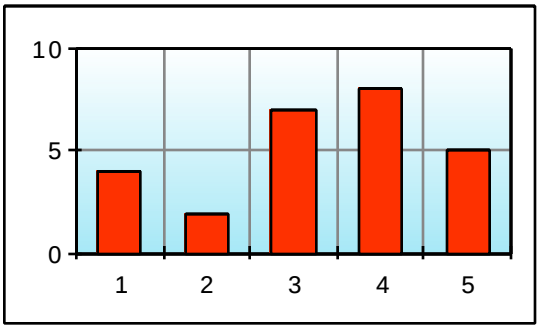

```
OpenDrawing(200;120)
     ChartData(4 2 7 2 4 9 6)
     RadarChart(oval)
     RadarChartOptions(0)
     MajorGridLineColors(all;all;black)
     ChartBackground(xy;255 220 220;;2;black;;2)
     Background(255 255 200;;1;;;3;lightGray)
CloseDrawing()
```



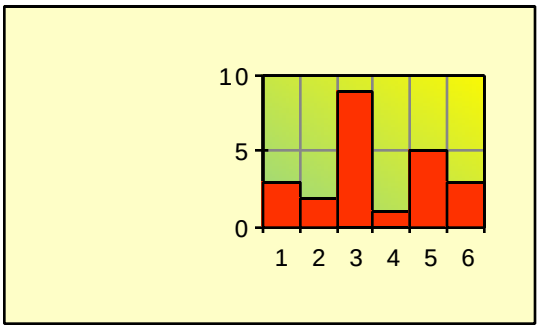
ChartBackgroundPict(ebenenindex;quelle;name)

Mit der Funktion `ChartBackgroundPict()` kann analog zur Funktion `ChartBackground()` ein auf das Diagrammraster begrenzter Bildhintergrund definiert werden. Das 1. Argument *ebenenindex* ist ident wie in Funktion `ChartBackground()`, die restlichen Argumente *quelle* und *name* sind gleich wie unter `BackgroundPict()` beschrieben. Beispiele:

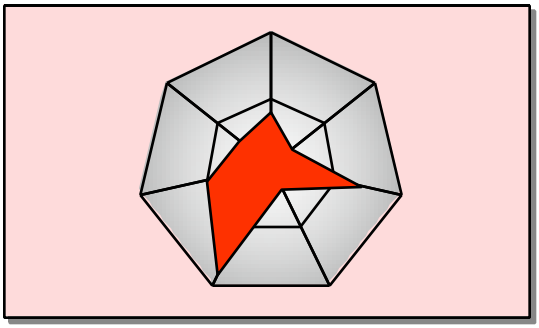
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     BarChart()
     GridFrame()
     ChartBackgroundPict(all;resource;"3")
CloseDrawing()
```



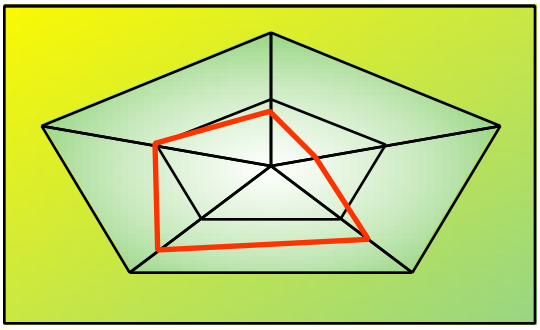

```
OpenDrawing(200;120)
     OpenView(70;10;120;100)
          ChartData(3 2 9 1 5 3)
          BarChart(;0)
          GridFrame()
          ChartBackgroundPict(xy;resource;"30")
     CloseView()
     Background(255 255 200)
CloseDrawing()
```




```
OpenDrawing(200;120)
     ChartData(4 2 7 2 9 5 3)
     RadarChart()
     RadarChartOptions(0;poly)
     MajorGridLineColors(all;all;black)
     ChartBackgroundPict(xy;resource;"34")
     Background(255 220 220;;1;black;;2;gray)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 2 7 8 5)
     RadarChart(oval)
     RadarChartOptions(0;poly)
     FillStyle(all;;transparent)
     BorderStyle(1;poly;2;red)
     MajorGridLineColors(all;all;black)
     ChartBackgroundPict(xy;resource;"38")
     BackgroundPict(resource;"29")
CloseDrawing()
```



Raster

Zur Gestaltung des Rasters stehen insgesamt 12 Funktionen zur Verfügung. Diese ermöglichen die Gestaltung von:

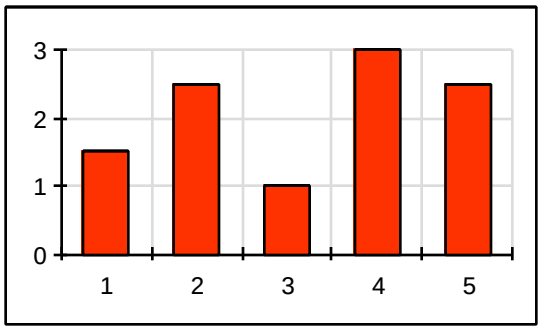
- Rasterlinien und Rasterflächen
- Grob- und Feinraster
- Rasterrahmen

Standardmäßig werden 1 Pixel breite, graue Rasterlinen verwendet.

MajorGridLineWidths(richtungsachse;verteilungsachse;
strichstärke1;strichstärke2...)
MajorGridLineColors(richtungsachse;verteilungsachse;
farbe1;farbe2...)
MajorGridLinePatterns(richtungsachse;verteilungsachse;
muster1;muster2...)

Rasterlinien können unterschiedlich in Farbe, Muster und Linienstärke definiert werden. Ist die Anzahl der Rasterlinien größer als die Anzahl der definierten Strichstärken, Farben oder Muster, so werden diese periodisch wiederholt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart()
     MajorGridLineColors(all;all;lightGray)
CloseDrawing()
```



Rasterlinien können durch Nullsetzen der Strichstärke unterdrückt werden. Falls weder die Richtungsachse noch die Verteilungsachse vorgegeben werden, beziehen sich die Funktionen auf alle Raster, d.h. bei zweidimensionalen kartesischen Diagrammen, sowohl auf das horizontale als auch auf das vertikale Raster. Der gezielte Zugriff auf eine spezielle Rasterrichtung erfolgt durch die beiden Argumente *richtungsachse* und *verteilungsachse*.

Bei 2-dimensionalen, kartesischen Diagrammen:

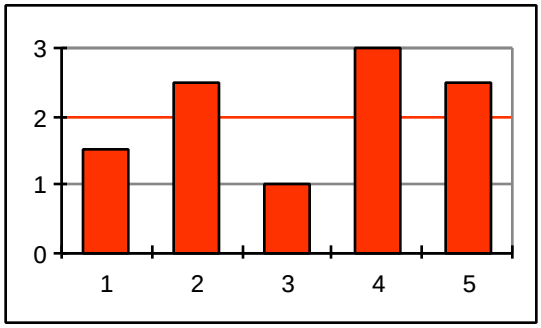
horizontales Raster: richtungsachse = x, verteilungsachse = y
vertikales Raster: richtungsachse = y, verteilungsachse = x

Bei 2-dimensionalen, polaren Diagrammen:

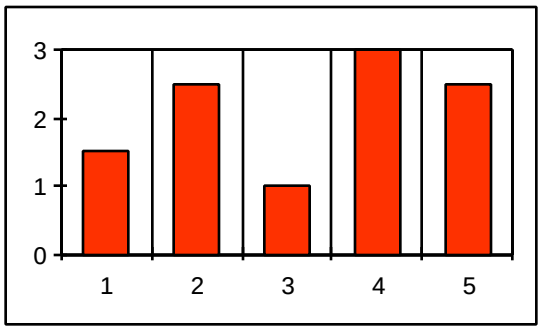
radiales Raster: richtungsachse = x, verteilungsachse = y
konzentr. Raster: richtungsachse = y, verteilungsachse = x

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart()

     // horizontale Rasterlinien
     MajorGridLineColors(x;y;gray;gray;red)

     // vertikale Rasterlinien ausblenden
     MajorGridLineWidths(y;x;0)

     // 1-Pixel breiter grauer Rahmen
     GridFrame(xy;1;gray)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart()
     MajorGridLineColors(all;all;black)

     // horizontale Rasterlinien ausblenden
     MajorGridLineWidths(x;y;0)

     // 1-Pixel breiter schwarzer Rahmen
     GridFrame()
CloseDrawing()
```

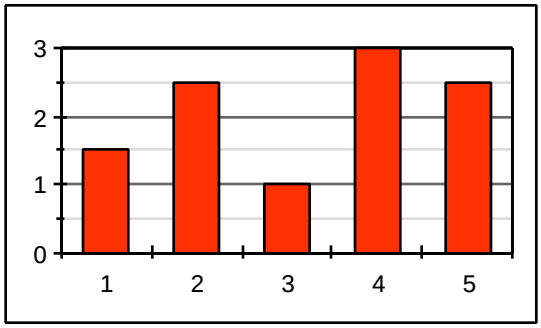


MinorGridLineWidths(richtungsachse;verteilungsachse;
strichstärke1;strichstärke2...)
MinorGridLineColors(richtungsachse;verteilungsachse;
farbe1;farbe2...)
MinorGridLinePatterns(richtungsachse;verteilungsachse;
muster1;muster2...)

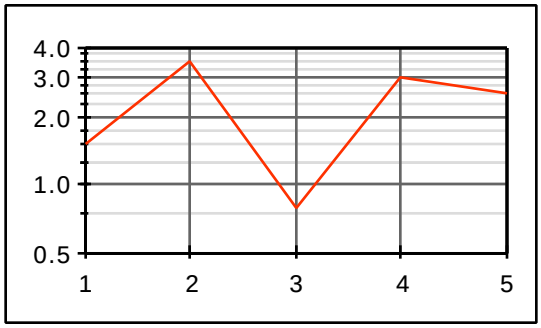
Die Funktionsweise des Feinraster ist gleich dem Grobraster. Zu beachten ist, dass das Feinraster erst durch Vorgabe einer Feinintervallanzahl größer 1 in der Funktion `Scaling()` — siehe Abschnitt *Achsen* — aktiviert wird.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart()

     // 3 Grobintervalle, 2 Feinintervalle
     Scaling(y;linear;;;3;2)

     // Grobraster
     MajorGridLineColors(all;all;darkGray)
     // vertikale Rasterlinien ausblenden
     MajorGridLineWidths(y;x;0)

     // Feinraster
     MinorGridLineColors(all;all;lightGray)

     // 1-Pixel breiter schwarzer Rahmen
     GridFrame()
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 3.5 0.8 3 2.5)
     LineChart()
     Scaling(y;log;;;3;4)

     // Grobraster
     MajorGridLineColors(all;all;darkGray)

     // Feinraster
     MinorGridLineColors(all;all;lightGray)

     // 1-Pixel breiter schwarzer Rahmen
     GridFrame()
CloseDrawing()
```

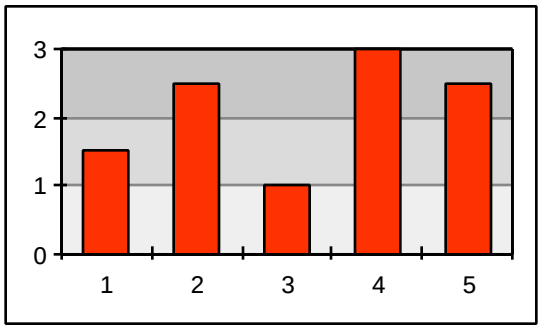


MajorGridStripeColors(richtungsachse;verteilungsachse;
farbe1;farbe2...)
MajorGridStripePatterns(richtungsachse;
verteilungsachse;
muster1;muster2...)
MinorGridStripeColors(richtungsachse;verteilungsachse;
farbe1;farbe2...)
MinorGridStripePatterns(richtungsachse;
verteilungsachse;
muster1;muster2...)

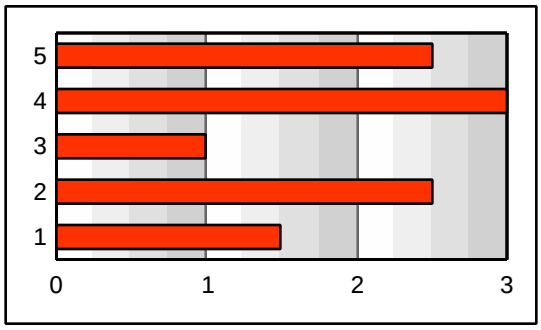
Rasterstreifen können optional anstelle von Rasterlinien oder auch in Kombination mit Rasterlinien verwendet werden. Der Aufbau der Funktionen und deren Handhabung ist gleich den Rasterlinien-Funktionen. Bei Polar- und Radardiagrammen werden Rasterstreifen nicht unterstützt.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart()

     // horizontale Rasterstreifen
     MajorGridStripeColors(x;y;240 240 240;
                               220 220 220;
                               200 200 200)

     // vertikale Rasterlinien ausblenden
     MajorGridLineWidths(y;x;0)

     // 1-Pixel breiter schwarzer Rahmen
     GridFrame()
CloseDrawing()
```



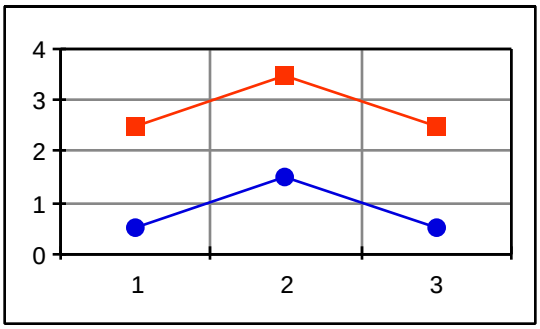
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1 3 2.5)
     BarChart(horizontal)
     Scaling(x;linear;0;3;3;4)

     // Alle Marker ausblenden
     AxisMajorTicks(all;0)
     AxisMinorTicks(all;0)

     // vertikale Rasterlinien
     MajorGridLineColors(y;x;darkGray)

     // vertikale Rasterstreifen
     MinorGridStripeColors(y;x;255 255 255;
                               240 240 240;
                               225 225 225;
                               210 210 210)

     // Rasterlinien ausblenden
     MajorGridLineWidths(x;y;0)
     MinorGridLineWidths(y;x;0)

     // 1-Pixel breiter schwarzer Rahmen
     GridFrame()
CloseDrawing()
```



GridFrame(ebenenindex;strichstärke;farbe;muster)

Durch die Funktion `GridFrame()` besteht die Möglichkeit, einen Rahmen um das Raster zu ziehen. Dabei kann die Breite, Farbe und das Muster des Rahmens kontrolliert werden. Als 1. Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Beispiel:

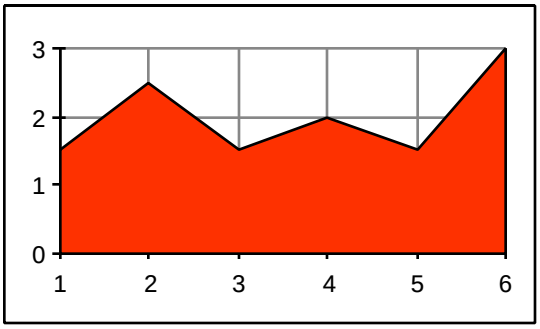
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(2.5 3.5 2.5; 0.5 1.5 0.5)
     LineChart(symbol;on)
     SymbolStyle(1;square;6)
     SymbolStyle(2;bullet;6)
     // 1-Pixel breiter schwarzer Rahmen
     GridFrame(xy;1;black)
CloseDrawing()
```



GridLocation(ebenenindex;rasteranordnung)

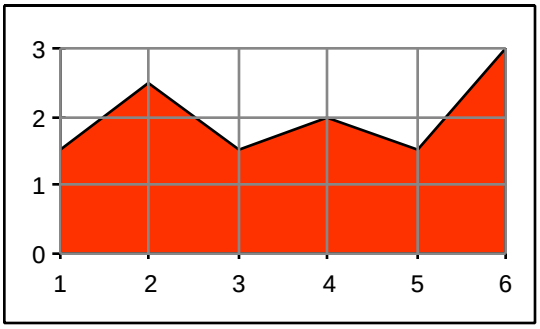
Standardmäßig wird das Raster stets im Hintergrund eines Diagramms dargestellt (*rasteranordnung=back*). Als 1. Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.5 2 1.5 3)
     AreaChart()
     GridLocation(xy;back)
CloseDrawing()
```



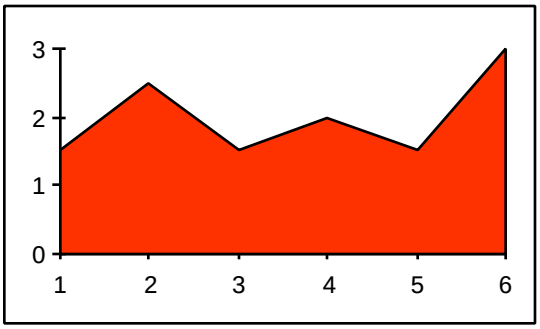
Durch *rasteranordnung=front* kann das Raster im Vordergrund platziert werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.5 2 1.5 3)
     AreaChart()
     GridLocation(xy;front)
CloseDrawing()
```



Optional können Raster durch *rasteranordnung=none* auf einfache Weise ausgeblendet werden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1.5 2.5 1.5 2 1.5 3)
     AreaChart()
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```


3
2
1
0
1
2
3
4
5
6

Achsen

Zur Gestaltung der Achsen stehen insgesamt 18 Funktionen zur Verfügung. Diese ermöglichen:

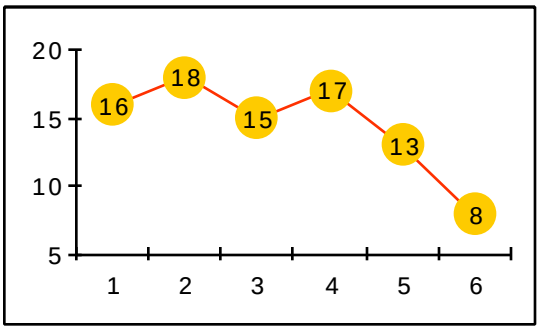
- die Festlegung unterschiedlicher Skalierungen
- die Gestaltung der Achsenlinien und Skalenmarkierungen
- ein flexibles Gestalten und Platzieren der Achsenbeschriftungen
- die Feinabstimmung mittels zahlreicher Optionen

Bei allen 18 Funktionen wird als erstes Argument der Achsenindex festgelegt. Dazu stehen die vier Konstanten *x*, *y*, *z* und *all* zur Verfügung. Wird kein Achsenindex vorgegeben oder *achsenindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle Achsen, d.h. zum Beispiel bei zweidimensionalen Diagrammen auf die x- und y-Achse.

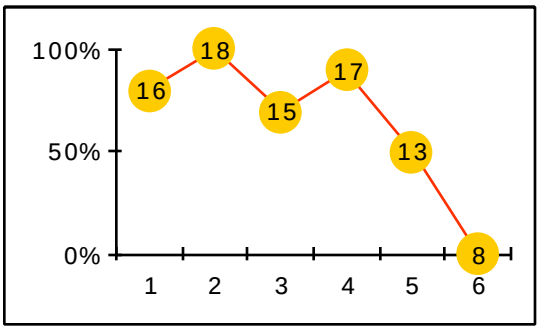
Scaling(achsenindex;typ;minWert;maxWert;
grobintervallAnzahl;feinintervallAnzahl;
basis;äquidistanteLogSkala)

Mit der Skalierungsfunktion `Scaling()` kann die Art der Skalierung, sowie die Anzahl der Unterteilungen und der Wertebereich festgelegt werden. Wird keine Skalierungsfunktion angeführt, so wird automatisch eine passende lineare Skalierung berechnet.
Das 2. Argument *typ* ermöglicht die Wahl zwischen linearer (*typ=linear*), prozentueller (*typ=percent*) und logarithmischer Skalierung (*typ=log*). Wird kein Typ vorgegeben, so wird eine linear Skalierung verwendet. Beispiele:

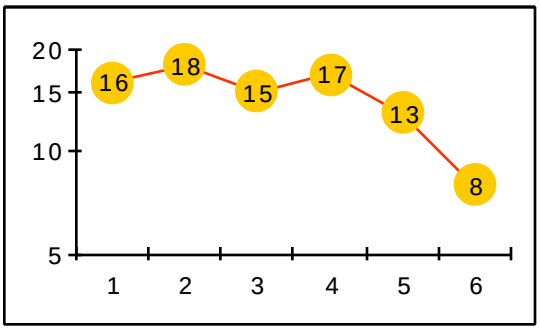
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol+label;on)
     Scaling(y;linear) // entspricht autom. Skalierung
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     GridLocation(all;none)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol+label;on)
     Scaling(y;percent)
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     GridLocation(all;none)
CloseDrawing()
```

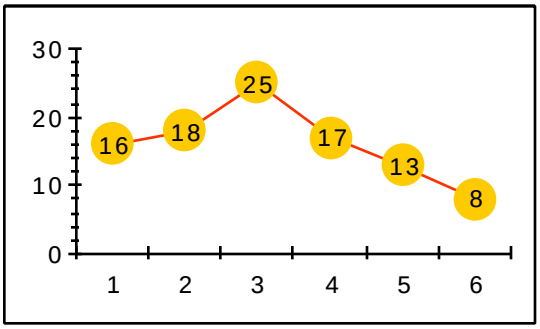



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol+label;on)
     Scaling(y;log)
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     GridLocation(all;none)
CloseDrawing()
```

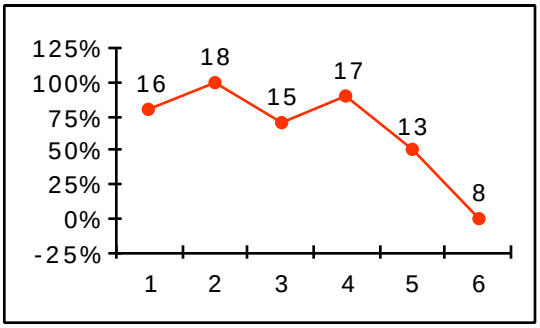


Durch das 3. und 4. Argument kann ein minimaler und maximaler Skalenwert vorgegeben werden. Wird der Minimalwert oder der Maximalwert nicht vorgegeben, so wird automatisch ein passender Wert gesucht.
Mit dem 5. und 6. Argument kann die Anzahl der Grob- und Feinintervalle kontrolliert werden. Die Anzahl der Feinintervalle bezieht sich dabei auf die Anzahl der Intervalle innerhalb eines Grobintervalls, d.h. sollen zum Beispiel die Grobintervalle durch fünf Feinintervalle unterteilt werden, so ist *feinintervallAnzahl=5* zu setzen; sollen keine Feinmarkierungen verwendet werden, so ist *feinintervallAnzahl=1* (default). Falls die Anzahl der Grobintervalle nicht vorgegeben wird, wird die Skala standardmäßig in vier Grobintervalle unterteilt. Beispiele:

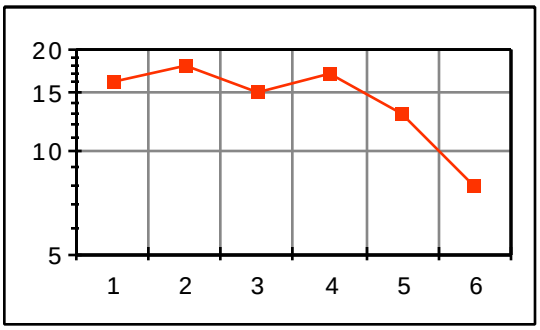
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 25 17 13 8)
     LineChart(symbol+label;on)
     Scaling(y;linear;0;30;3;5) // 3 Grob- u. 5 Feininter.
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     GridLocation(all;none)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol+label;on)
     Scaling(y;percent;-25;125;6)
     SymbolStyle(1;bullet;4;;red)
     LabelOptions(all;topCenter)
     GridLocation(all;none)
CloseDrawing()
```

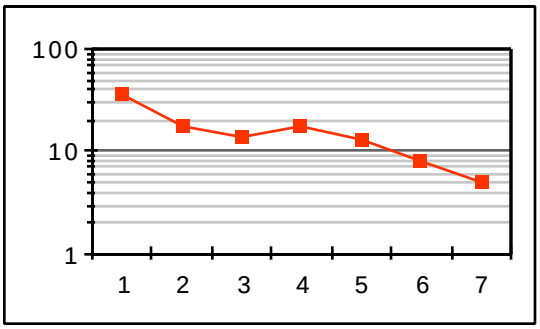



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 15 17 13 8)
     LineChart(symbol;on)
     Scaling(y;log;5;20;3;5) // 3 Grob- u. 5 Feininter.
     SymbolStyle(1;square;4;;red)
     MinorGridLineWidths(all;all;0) // kein Feinraster
     GridFrame()
CloseDrawing()
```

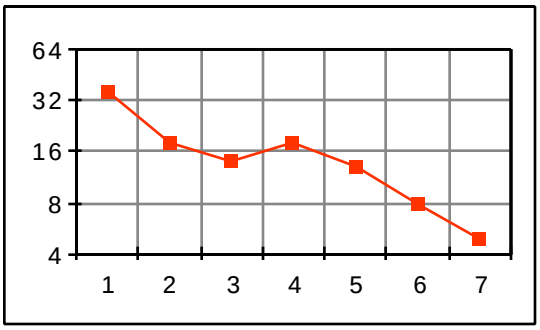


Die Argumente *basis* und *äquidistanteLogSkala* finden ausschließlich bei einer logarithmischen Skalierung Verwendung. Mit dem Argument *basis* wird die Basis des Logarithmus festgelegt. Falls keine Basis vorgegeben wird, wird der dekadische Logarithmus (*basis=10*) zur Skalierung verwendet. Durch Aktivieren des Arguments *äquidistanteLogSkala=on* finden nur Skalenwerte Verwendung, welche ein Vielfaches der Basis sind, zum Beispiel bei einer logarithmischen Skalierung zur Basis 10, werden nur Werte wie 0.01, 0.1, 1, 10, 100, 1000 usw. verwendet. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(36 18 14 18 13 8 5)
     LineChart(symbol;on)
     Scaling(y;log;;;;9;10;on) // 9 Feinintervalle
     SymbolStyle(1;square;4;;red)
     MajorGridLineWidths(y;x;0) // kein vertikales Raster
     MajorGridLineColors(;;100 100 100)
     MinorGridLineColors(;;200 200 200)
     GridFrame()
CloseDrawing()
```



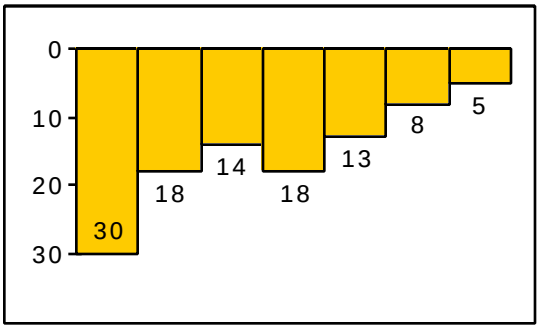

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(36 18 14 18 13 8 5)
     LineChart(symbol;on)
     Scaling(y;log;;;;;2;on) // Log. zur Basis 2
     SymbolStyle(1;square;4;;red)
     GridFrame()
CloseDrawing()
```



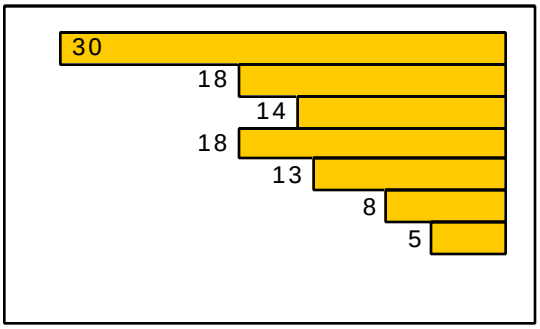
ScalingOptions(achsenindex;umgekehrteSkala; nurGanzzahlen)

Durch Aktivieren des 2. Arguments *umgekehrteSkala=on* kann die Richtung der Skalierung umgedreht werden, d.h. die Skalierung der x-Achse verläuft nicht mehr aufsteigend von links nach rechts, sondern umgekehrt, aufsteigend von rechts nach links, die y-Achse entsprechend, aufsteigend von oben nach unten. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(30 18 14 18 13 8 5)
     BarChart(label;0)
     FillStyle(1;darkYellow)
     ScalingOptions(y;on)   // y-Skala umkehren
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

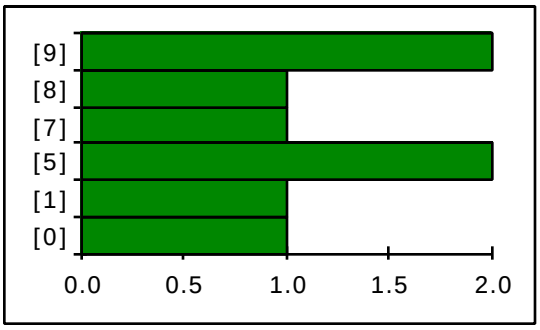


```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(30 18 14 18 13 8 5)
     BarChart(horizontal+label;0)
     FillStyle(1;darkYellow)
     ScalingOptions(all;on) // x- und y-Skala umkehren
     AxisOptions(all;none)  // keine Achsen
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

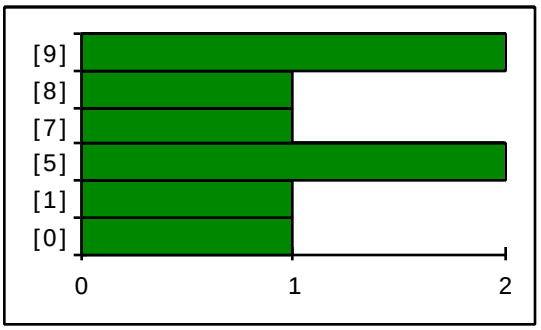


Durch Aktivieren des 3. Arguments *nurGanzzahlen=on* können nicht-ganzzahlige Skalierungswerte ausgeblendet werden. Dies erweist sich zweckmässig zur Darstellung ganzzahliger Größen, wie zum Beispiel bei Häufigkeitsverteilungen. Beispiel:

```
// ohne ScalingOptions()
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 9 0 9 5 5 7 8)
     Histogram(horizontal)
     HistogramOptions(on)  // nach Histogram()!
     FillStyle(all;green)
     GridLocation(xy;none)
CloseDrawing()
```



```
// mit ScalingOptions()
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 9 0 9 5 5 7 8)
     Histogram(horizontal)
     HistogramOptions(on)  // nach Histogram()!
     FillStyle(all;green)
     GridLocation(xy;none)
     ScalingOptions(x;;on) // nur ganzahlige x-Skala
CloseDrawing()
```

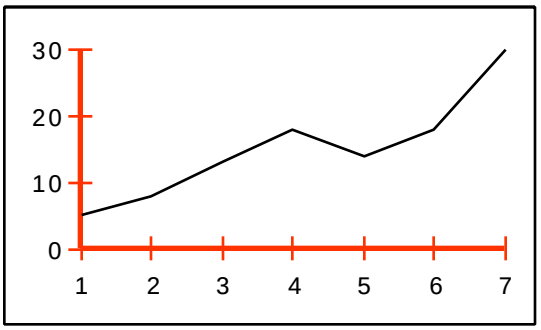


AxisLine(achsenindex;strichstärke;farbe;muster)
AxisMajorTicks(achsenindex;länge;strichstärke;farbe;muster;platzierung)
AxisMinorTicks(achsenindex;länge;strichstärke;farbe;muster;platzierung)

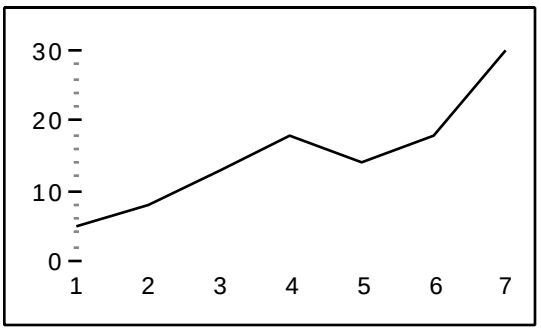
Durch die Funktion `AxisLine()` kann das Aussehen der Achsenlinien kontrolliert werden, durch die beiden Funktionen `AxisMajorTicks()` und `AxisMinorTicks()` das Aussehen der Grob- und Feinmarkierungen. Standardmäßig werden Achsenlinien und Skalenmarkierungen in schwarz und in einer Breite von 1 Pixel dargestellt. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Zusätzlich können bei den Grob- und Feinmarkierungen mit dem Argument *länge* die Markierungslänge in Pixel und mit dem Argument *platzierung* die Ausrichtung der Markierungen kontrolliert werden. Markierungen können entweder innerhalb des Diagramms platziert werden (*platzierung=in*), außerhalb des Diagramms (*platzierung=out*) oder zur Hälfte innerhalb und zur Hälfte außerhalb (*platzierung=center*). Die Grobmarkierungen sind standardmäßig 5 Pixel lang, die Feinmarkierungen 3 Pixel.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 8 13 18 14 18 30)
     LineChart()
     LineStyle(1;poly;1;black)
     AxisLine(all;2;red)
     AxisMajorTicks(all;9;1;red)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



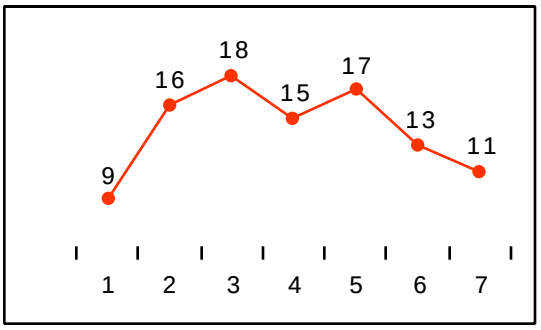

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 8 13 18 14 18 30)
     LineChart()
     Scaling(y;linear;0;30;3;5) // 3 Grob- u. 5 Feininter.
     LineStyle(1;poly;1;black)
     AxisLine(all;0)        // keine Achsenlinien
     AxisMajorTicks(x;0)    // keine x-Skalenmarkierungen
     AxisMajorTicks(y;5)
     AxisMinorTicks(y;2;1;gray)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



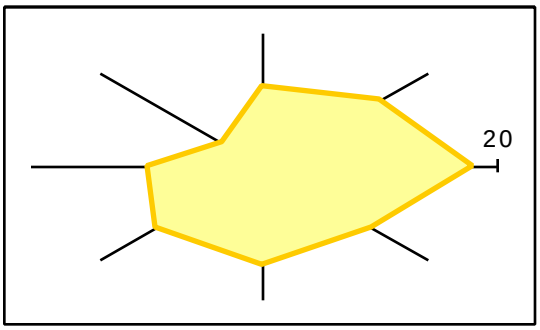
AxisOptions(achsenindex;achsenanordnung; achseVerschieben)

Mit dem 2. Argument *achsenanordnung* können Achsen entweder hinter dem Diagramm angeordnet werden (*achsenanordnung=back*) oder davor (*achsenanordnung=front*). Letztere Anordnung erweist sich zum Beispiel zweckmäßig bei Polar- und Radardiagrammen. Optional können Achsen durch *achsenanordnung=none* komplett ausgeblendet werden. Beispiele:

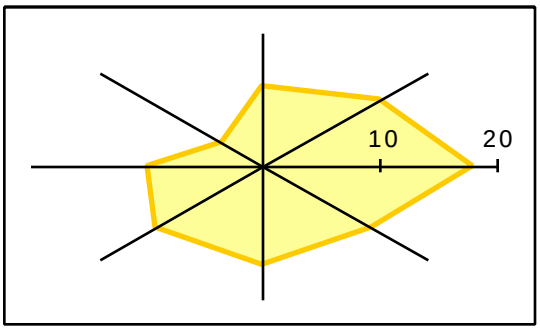
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 16 18 15 17 13 11)
     LineChart(symbol+label;on)
     SymbolStyle(1;bullet;4;;red)
     LabelOptions(all;topCenter)
     AxisOptions(y;none)    // keine y-Achse
     AxisLine(x;0)          // x-Achsenlinie ausblenden
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(18 13 15 13 10 5 12 14)
     RadarChart(oval;90)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     AxisOptions(all;back) // Achsen im Hintergr.(default)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

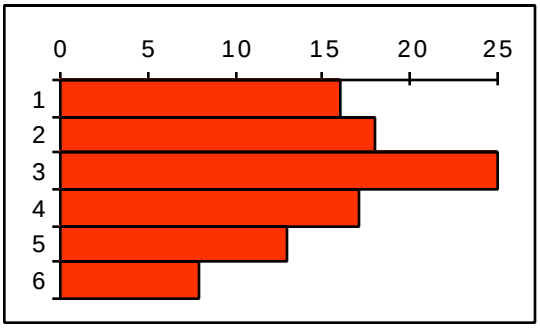



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(18 13 15 13 10 5 12 14)
     RadarChart(oval;90)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     AxisOptions(all;front) // Achsen in den Vordergrund
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

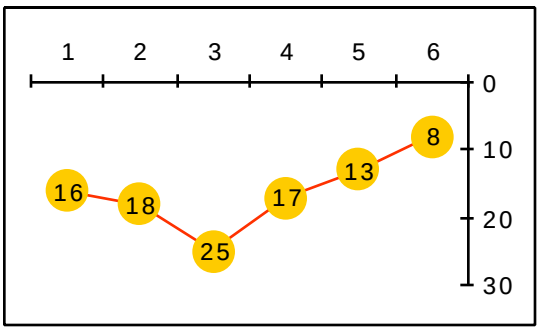


Durch Aktivieren des 3. Arguments *achseVerschieben=on* können Achsen auf die gegenüberliegende Seite verschoben werden, d.h. die x-Achse wird von unten nach oben verschoben, die y-Achse von links nach rechts. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 25 17 13 8)
     BarChart(horizontal;0)
     ScalingOptions(y;on)   // y-Skalierung umkehren
     AxisOptions(x;;on)     // x-Achse verschieben
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 25 17 13 8)
     LineChart(symbol+label;on)
     ScalingOptions(y;on) // y-Skalierung umkehren
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     AxisOptions(all;;on) // x- und y-Achse verschieben
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

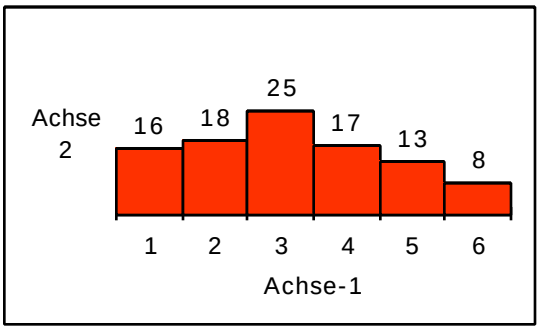


AxisLabelText(achsenindex;text1;text2...)

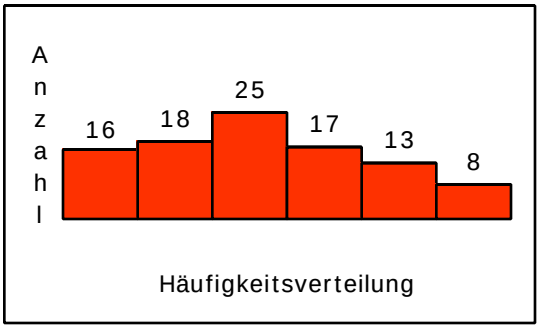
Durch die Funktion `AxisLabelText()` können Achsen mit einer Beschriftung versehen werden. Dabei sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "\n" möglich. Enthält der Text eine Formatanweisung, wie zum Beispiel "|u|", wird der Achsenindex ausgegeben, d.h. bei der x-Achsenbeschriftung "1", bei der y-Achsenbeschriftung "2". Die Handhabung von Texten in Verbindung mit Formatanweisungen wird im Abschnitt *Stile*, Funktion `LabelTexts()` erläutert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 25 17 13 8)
     BarChart(label;0)
     AxisLine(y;0)                  // keine y-Achsenlinie
     AxisMajorTicks(y;0)            // keine y-Markierungen
     AxisMajorTickLabelTexts(y;"") // keine y-Beschriftung
     AxisLabelText(x;"Achse-|u|")
     AxisLabelText(y;"Achse\n|u|")
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

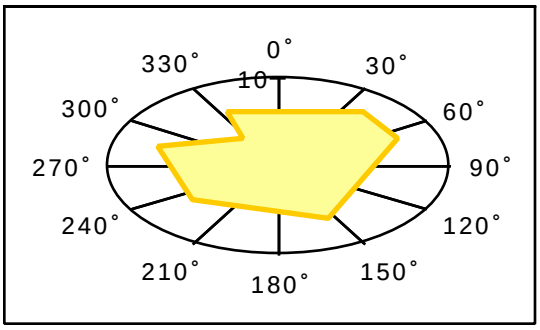



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 25 17 13 8)
     BarChart(label;0)
     AxisLine(all;0)                  // keine Achsenlinie
     AxisMajorTicks(all;0)            // keine Markierungen
     AxisMajorTickLabelTexts(all;"") // keine Skalenbesch.
     AxisLabelText(x;"Häufigkeitsverteilung")
     AxisLabelText(y;"A\nn\nz\na\nh\nl")
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

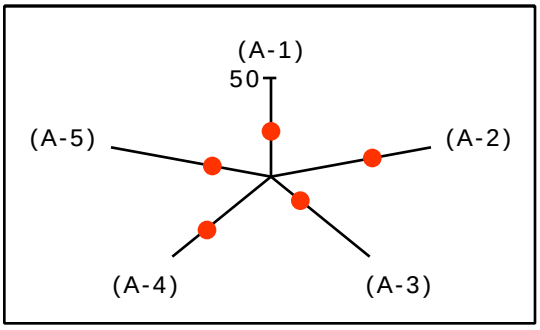


Bei Polar- und Radardiagrammen können auch mehrere Texte übergeben werden, die den einzelnen Achsen entsprechend zugeordnet werden. Ist die Anzahl der übergebenen Texte niedriger als die Anzahl der Achsen, so werden die Texte periodisch wiederholt. Enthält ein Text eine Formatanweisung, so wird bei Polardiagrammen der Winkel in Grad ausgegeben, bei Radardiagrammen der Achsenindex. Zu beachten ist, dass bei Polar- und Radardiagrammen der *achsenindex=1* oder *achsenindex=x* zu setzen ist, alle anderen Achsenindizes werden ignoriert. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(5 7  3 -5 -7 -2 -3; // x-Werte
               6 3 -6 -4  2  3  6) // y-Werte
     PolarChart(oval)
     FillStyle(1;lightYellow)
     BorderStyle(1;poly;2;darkYellow)
     MajorGridLineColors(all;all;black)
     AxisLabelText(all;"|u|°")
CloseDrawing()
```



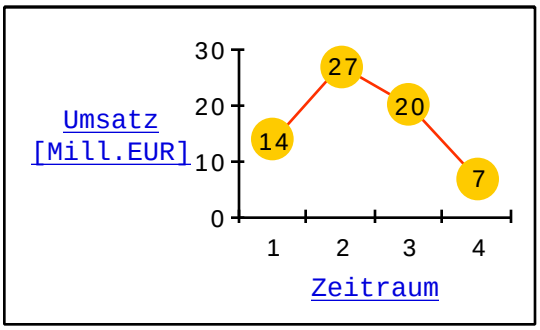

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 32 15 33 18)
     RadarChart(oval+symbol)
     FillStyle(;;transparent)
     BorderStyle(;none)
     SymbolStyle(1;bullet;6;1;red)
     AxisLabelText(1;"(A-|u|)")
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



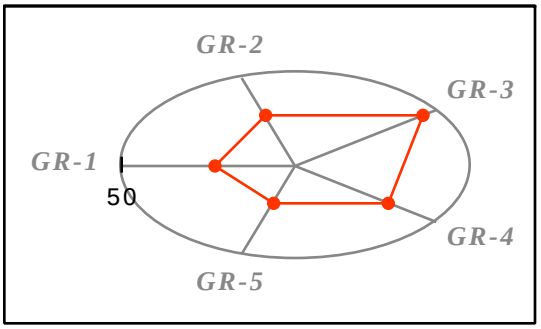
AxisLabelStyle(achsenindex;schrift;größe;stil;farbe; ausrichtung)

Mit der Funktion `AxisLabelStyle()` kann der Achsenbeschriftung ein Schriftstil zugeordnet werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(14 27 20 7)
     LineChart(symbol+label;on)
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     AxisLabelText(x;"Zeitraum")
     AxisLabelText(y;"Umsatz\n[Mill.EUR]")
     AxisLabelStyle(all;"Courier";10;underline;blue)
     GridLocation(all;none)
CloseDrawing()
```



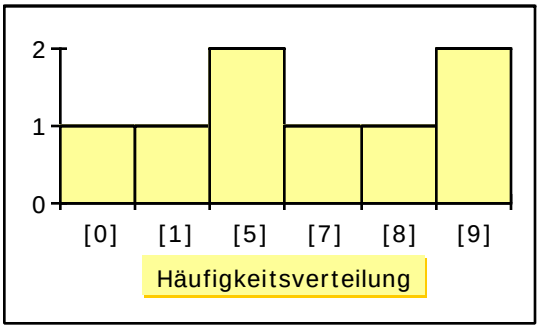
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 29 45 33 21)
     RadarChart(oval+symbol;-90)
     FillStyle(all;;transparent)
     BorderStyle(1;poly;1;red)
     SymbolStyle(1;bullet;4;1;red)
     AxisLine(1;0)
     AxisLabelText(1;"GR-|u|")
     AxisLabelStyle(1;"Times";10;bold+italic;gray)
CloseDrawing()
```



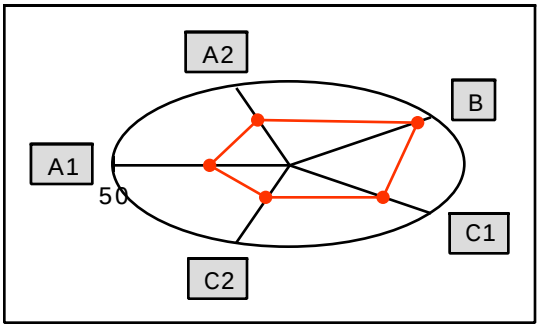
AxisLabelBackground(achsenindex;füllfarbe;füllmuster;
rahmenbreite;rahmenfarbe;rahmenmuster;
schattenabstand;schattenfarbe;schattenmuster)

Durch die Funktion `AxisLabelBackground()` kann der Hintergrund der Achsenbeschriftung gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 9 0 9 5 5 7 8)
     Histogram()
     HistogramOptions(on)  // nach Histogram()!
     FillStyle(all;lightYellow)
     ScalingOptions(y;;on) // nur ganzahlige y-Skala
     AxisLabelText(x;" Häufigkeitsverteilung ")
     AxisLabelBackground(x;lightYellow;;0;;;1;darkYellow)
     GridLocation(xy;none)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 29 45 33 21)
     RadarChart(oval+symbol;-90)
     FillStyle(all;;transparent)
     BorderStyle(1;poly;1;red)
     SymbolStyle(1;bullet;4;1;red)
     AxisLine(1;0)
     AxisLabelText(1;" A1 ";" A2 ";" B ";" C1 ";" C2 ")
     AxisLabelBackground(x;lightGray;;1)
     MajorGridLineColors(all;all;black)
CloseDrawing()
```



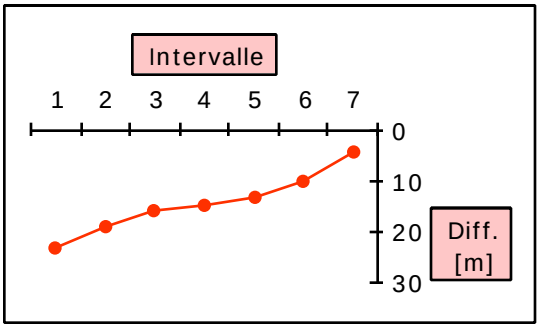
AxisLabelOptions(achsenindex;platzierung;hVersatz; vVersatz)

Durch die Funktion `AxisLabelOptions()` kann die Platzierung der Achsenbeschriftung kontrolliert werden. Das Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen:

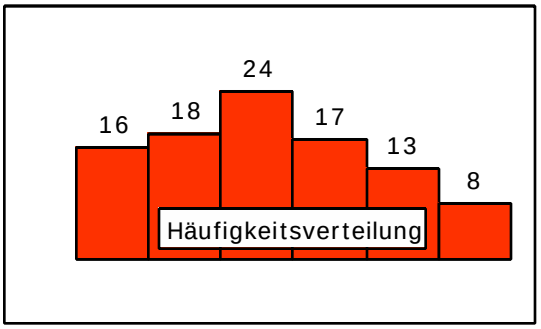
Konstante	*Wert*	*Anmerkung*
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Standardmäßig wird die Beschriftung der x-Achse in der Mitte unterhalb der Achsenlinie platziert (*platzierung=bottomCenter*), die Beschriftung der y-Achse in der Mitte links von der Achsenlinie (*platzierung=centerLeft*). Bei Polar- und Radardiagrammen wird die Achsenbeschriftung immer außerhalb des Diagramms platziert, das Argument *platzierung* wird ignoriert. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 19 16 15 13 10 4)
     LineChart(symbol;on)
     SymbolStyle(1;bullet;4)
     ScalingOptions(y;on) // y-Skalierung umkehren
     AxisOptions(all;;on) // x- und y-Achse verschieben
     AxisLabelText(x;" Intervalle ")
     AxisLabelText(y;" Diff. \n[m]")
     AxisLabelBackground(all;255 200 200)
     AxisLabelOptions(x;topCenter)
     AxisLabelOptions(y;bottomRight)
     GridLocation(all;none)
CloseDrawing()
```



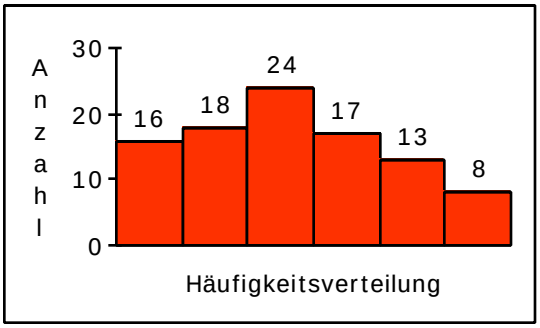

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 24 17 13 8)
     BarChart(label;0)
     AxisMajorTicks(x;0)           // keine x-Markierungen
     AxisMajorTickLabelTexts(x;"") // keine x-Skalenbesch.
     AxisOptions(y;none)
     AxisLabelText(x;"Häufigkeitsverteilung")
     AxisLabelBackground(x)
     AxisLabelOptions(x;topCenter)
     AxisOptions(x;front)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



Mit den Argumenten *hVersatz* und *vVersatz* kann die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben.

Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(16 18 24 17 13 8)
     BarChart(label;0)
     AxisLabelText(x;"Häufigkeitsverteilung")
     AxisLabelText(y;"A\nn\nz\na\nh\nl")
     AxisMajorTicks(x;0)           // keine x-Markierungen
     AxisMajorTickLabelTexts(x;"") // keine x-Skalenbesch.
     AxisLabelOptions(x;;;-10)     // 10 Pixel nach oben
     AxisLabelOptions(y;;-6)       //  6 Pixel nach links
     GridLocation(all;none)        // kein Raster
CloseDrawing()
```

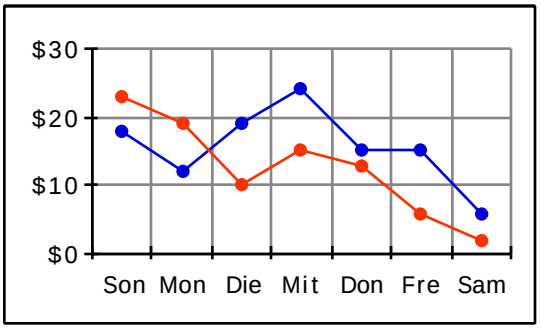


AxisMajorTickLabelTexts(achsenindex;text1;text2...)

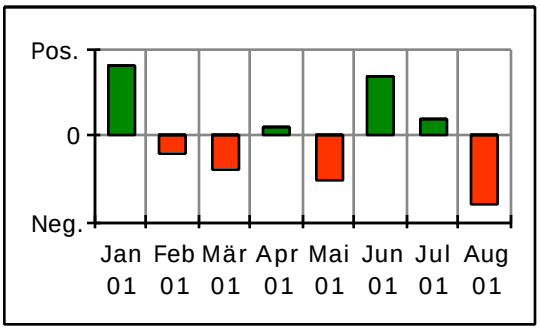
Die Beschriftung der Skalenmarkierungen besteht standardmäßig aus den jeweiligen Skalenwert. Durch die Funktion `AxisMajorTickLabelTexts()` kann der Beschriftungstext der Grobmarkierungen kontrolliert werden — so ist es möglich, jeder Skalenmarkierung einen individuellen Text zuzuordnen. Die Texte werden periodisch wiederholt, falls die Anzahl der Grobmarkierungen größer ist als die Anzahl der übergebenen Texte. Enthält der Text eine Formatanweisung, wie zum Beispiel "|f1|", so wird der Skalenwert entsprechend formatiert ausgegeben. Die Handhabung von Texten in Verbindung mit Formatanweisungen wird im Abschnitt *Stile*, Funktion `LabelTexts()` erläutert. Mehrzeilige Texte sind durch Einfügen eines Zeilenumbruchs "\n" möglich.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 19 10 15 13  6 2;
               18 12 19 24 15 15 6)
     LineChart(symbol;on)
     SymbolStyle(1;bullet;4)
     SymbolStyle(2;bullet;4)
     AxisMajorTickLabelTexts(y;"$|u|")
     AxisMajorTickLabelTexts(x;"Son";"Mon";"Die";"Mit";
                               "Don";"Fre";"Sam")
CloseDrawing()
```



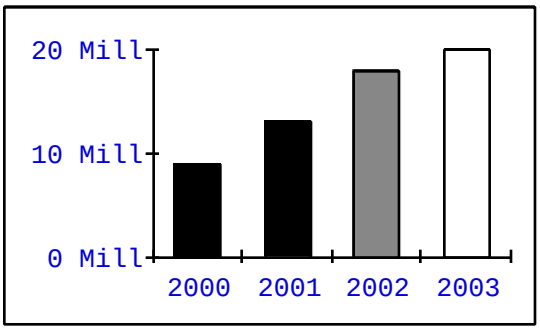

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(8 -2 -4 1 -5 7 2 -8)
     BarChart()
     BarChartOptions(;;on)
     Scaling(y;linear;-10;10;2)
     FillStyle(1;green)
     FillStyle(2;red)
     AxisMajorTickLabelTexts(y;"Neg.";"0";"Pos.")
     AxisMajorTickLabelTexts(x;"Jan\n01";"Feb\n01";
                               "Mär\n01";"Apr\n01";
                               "Mai\n01";"Jun\n01";
                               "Jul\n01";"Aug\n01")
CloseDrawing()
```



AxisMajorTickLabelStyle(achsenindex;schrift;größe;stil; farbe;ausrichtung)

Mit der Funktion AxisMajorTickLabelStyle() kann der Schriftstil der Skalenbeschriftung variiert werden. Beispiel:

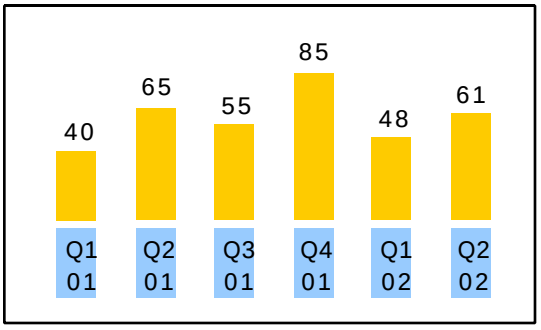
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 9000000 0 0 0;
                0 13000000 0 0;
                0 0 18000000 0;
                0 0 0 20000000)
     BarChart(stacked)
     FillStyle(all;black)
     FillStyle(3;gray)
     FillStyle(4;;transparent)
     AxisMajorTickLabelTexts(x;"2000";"2001";
                              "2002";"2003")
     AxisMajorTickLabelTexts(y;"|-6u| Mill")
     AxisMajorTickLabelStyle(all;"Courier";10;plain;blue)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



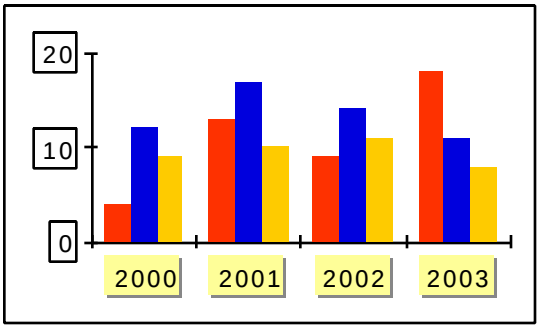
**AxisMajorTickLabelBackground(achsenindex;füllfarbe;
füllmuster;rahmenbreite;rahmenfarbe;
rahmenmuster;schattenabstand;
schattenfarbe;schattenmuster)**

Durch die Funktion AxisMajorTickLabelBackground() kann der Hintergrund der Skalenbeschriftung gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(40 65 55 85 48 61)
     BarChart(label)
     FillStyle(1;darkYellow)
     BorderStyle(1;none)
     LabelOptions(1;out)   // alle Beschriftungen außer-
                           // halb der Balken platzieren.
     AxisLine(all;0)       // keine Achsenlinien
     AxisMajorTicks(all;0) // kein Skalenmarkierungen
     AxisMajorTickLabelTexts(y;"") // keine y-Beschriftung
     AxisMajorTickLabelTexts(x;"Q1\n01";"Q2\n01";
                               "Q3\n01";"Q4\n01";
                               "Q1\n02";"Q2\n02")
     AxisMajorTickLabelBackground(x;lightBlue;;0)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 13  9 18;
               12 17 14 11;
                9 10 11  8)
     BarChart()
     FillStyle(3;darkYellow)
     BorderStyle(all;none)
     AxisMajorTickLabelTexts(x;"2000";"2001";
                             "2002";"2003")
     AxisMajorTickLabelBackground(x;lightYellow;;0;;;1)
     AxisMajorTickLabelBackground(y)
     GridLocation(all;none)
CloseDrawing()
```

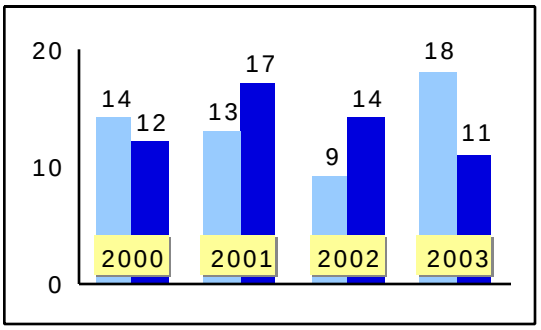


AxisMajorTickLabelOptions(achsenindex;platzierung; hVersatz;vVersatz;beschriftungJeMarker; beschriftungAbMarker)

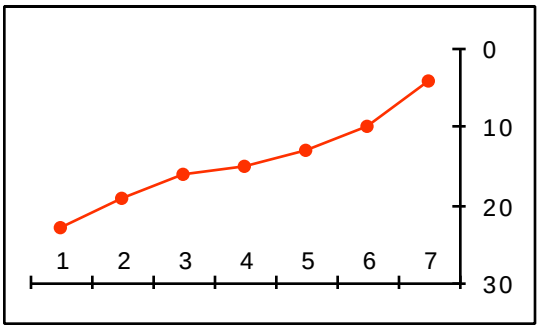
Die Funktion `AxisMajorTickLabelOptions()` dient zur Platzierung und Feinabstimmung der Skalenbeschriftung. Das Argument *platzierung* erlaubt die Positionierung der Beschriftung entweder innerhalb des Diagramms (*platzierung=in*) oder außerhalb des Diagramms (*platzierung=out*) oder auch direkt auf der Achsenlinie (*platzierung=center*). Mit den Argumenten *hVersatz* und *vVersatz* kann die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(14 13 9 18; 12 17 14 11)
     BarChart(label)
     FillStyle(1;lightBlue)
     BorderStyle(all;none)
     LabelOptions(all;out) // alle Beschriftungen außer-
                           // halb der Balken platzieren.
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     AxisMajorTickLabelTexts(x;"2000";"2001";
                              "2002";"2003")
     AxisMajorTickLabelBackground(x;lightYellow;;0;;;1)
     AxisMajorTickLabelOptions(x;in;;2) //2 Pxl nach unten
     AxisMajorTickLabelOptions(y;;-3)   //3 Pxl nach links
     AxisOptions(x;front) // Achsenbeschriftung vor Balken
     GridLocation(all;none)
CloseDrawing()
```

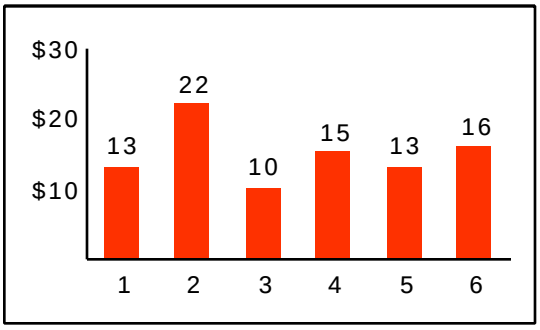



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(23 19 16 15 13 10 4)
     LineChart(symbol;on)
     SymbolStyle(1;bullet;4)
     ScalingOptions(y;on) // y-Skalierung umkehren
     AxisOptions(y;;on)   // y-Achse verschieben
     AxisMajorTickLabelOptions(x;in;;3) //3 Pxl nach unten
     AxisMajorTickLabelOptions(y;;3)   //3 Pxl nach rechts
     GridLocation(all;none)
CloseDrawing()
```

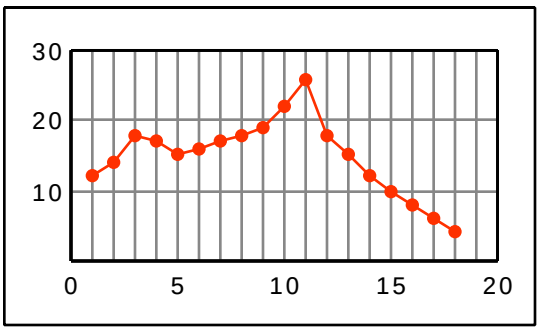


Durch die Argumente *beschriftungJeMarker* und *beschriftungAbMarker* können Skalenbeschriftungen ausgeblendet werden. So bewirken zum Beispiel die Argumente *beschriftungJeMarker=2* und *beschriftungAbMarker=3*, dass nur die 3. Markierung, 5. Markierung, 7. Markierung usw. dargestellt werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(13 22 10 15 13 16)
     BarChart(label)
     BorderStyle(all;none)
     AxisMajorTicks(all;0) // keine Skalenmarkierungen
     AxisMajorTickLabelTexts(y;"$|u|")
     AxisMajorTickLabelOptions(y;;;;1;2) //"$0" ausblenden
     GridLocation(all;none)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(12 14 18 17 15 16 17 18 19
               22 26 18 15 12 10 8 6 4)
     LineChart(symbol)
     SymbolStyle(all;bullet;4)
     Scaling(x;linear;0;20;20)
     AxisMajorTicks(all;0)
     AxisMajorTickLabelOptions(x;;;;5)   // jeder 5. Wert.
     AxisMajorTickLabelOptions(y;;;;1;2) // "0" ausblenden
     GridFrame()
CloseDrawing()
```

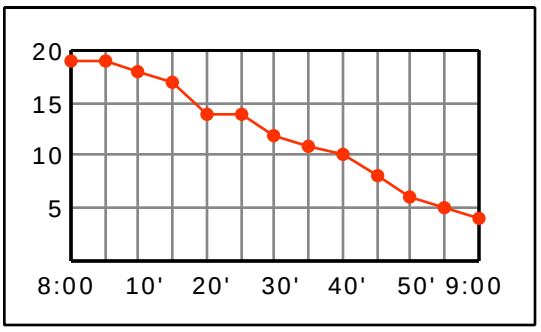


AxisMinorTickLabelTexts(achsenindex;text1;text2...)
AxisMinorTickLabelStyle(achsenindex;schrift;größe;stil;
farbe;ausrichtung)
AxisMinorTickLabelBackground(achsenindex;füllfarbe;
füllmuster;rahmenbreite;rahmenfarbe;
rahmenmuster;schattenabstand;
schattenfarbe;schattenmuster)
AxisMinorTickLabelOptions(achsenindex;platzierung;
hVersatz;vVersatz;
beschriftungJeMarker;
beschriftungAbMarker)

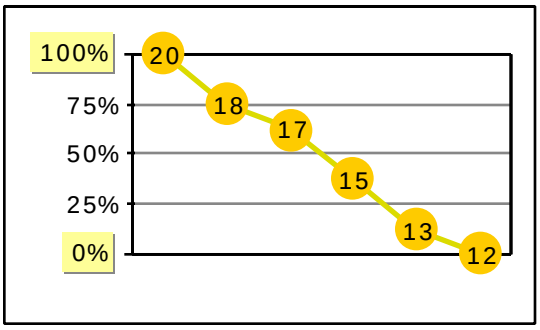
Die Funktionen zur Beschriftung der Feinmarkierungen sind ident aufgebaut wie die Funktionen zur Beschriftung der Grobmarkierungen.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 1  2  3  4  5  6  7  8  9 10 11 12 13;
               19 19 18 17 14 14 12 11 10  8  6  5  4)

     LineChart2D(symbol)
     Scaling(x;linear;1;13;1;12)
     SymbolStyle(1;bullet;4)

     // Achsen
     AxisMajorTicks(all;0) // keine Grobskalenmarkierungen
     AxisMinorTicks(all;0) // keine Feinskalenmarkierungen
     AxisMajorTickLabelTexts(x;"8:00";"9:00")
     AxisMajorTickLabelStyle(x;;;bold)
     AxisMinorTickLabelTexts(x;" 10';" 20';" 30';
                               " 40';" 50')
     AxisMinorTickLabelOptions(x;;;;2;2) // jeder 2. Wert.
     AxisMajorTickLabelOptions(y;;;;1;2) // "0" ausblenden

     // Raster
     MajorGridLinePatterns(all;all;black)
     MajorGridLineColors(all;all;gray)
     MinorGridLinePatterns(all;all;black)
     MinorGridLineColors(all;all;gray)
     GridFrame()
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(20 18 17 15 13 12)
     LineChart(symbol+label;on)
     Scaling(y;percent;0;100;1;4)
     LineStyle(1;poly;2;220 220 0))
     SymbolStyle(1;bullet;15;;darkYellow)
     LabelOptions(all;centerCenter)
     AxisOptions(x;none) // keine x-Achse
     AxisMajorTickLabelBackground(y;lightYellow;;0;;;1)
     AxisMinorTickLabelTexts(y;"|u|%")
     MajorGridLineWidths(y;x;0) // kein vertikales Raster
     GridFrame()
CloseDrawing()
```



Legende

Zur Gestaltung der Legende stehen vier Funktionen zur Verfügung. Diese ermöglichen:

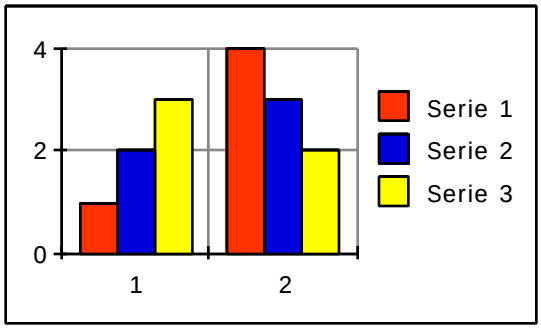
- das Definieren der Legendentexte
- die Festlegung des Schriftstils
- die Gestaltung des Hintergrundes
- das flexible Platzieren der Legende
- die Feinabstimmung mittels zahlreicher Optionen

Eine Legende wird nicht automatisch per default erstellt, sondern erst durch Aufrufen einer der vier nachstehend angeführten Funktionen.

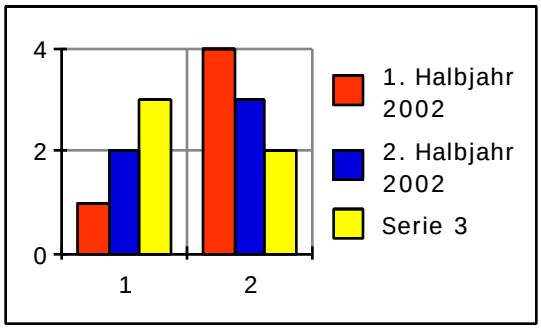
LegendTexts(text1;text2...)

Durch die Funktion `LegendTexts()` kann jeder Datenserie ein Namen zugeordnet werden. Dabei sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "\n" möglich. Falls keine oder zu wenige Legendentexte zur Verfügung gestellt werden, wird für die fehlenden Namen die Standardbezeichnung "Serie" plus einer laufenden Nummer verwendet. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts()
CloseDrawing()
```



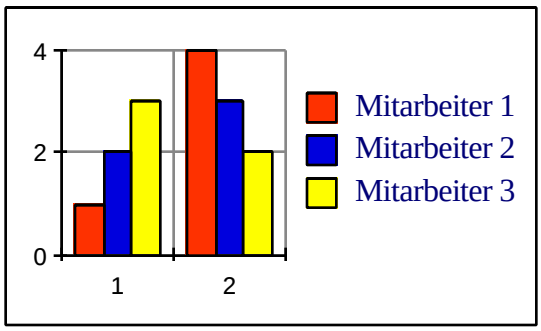
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("1. Halbjahr\n2002";
                 "2. Halbjahr\n2002")
CloseDrawing()
```



LegendStyle(schrift;größe;stil;farbe;ausrichtung)

Mit der Funktion `LegendStyle()` wird der Schriftstil der Legende festgelegt. Zum Beispiel:

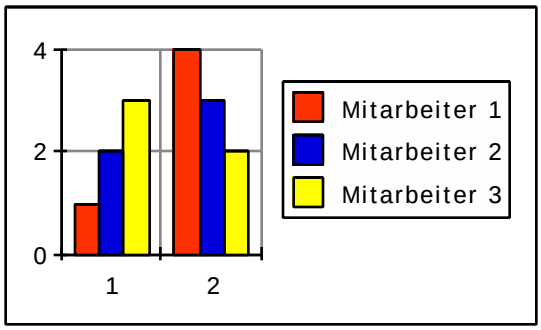
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendStyle("Times";12;plain;darkBlue)
CloseDrawing()
```



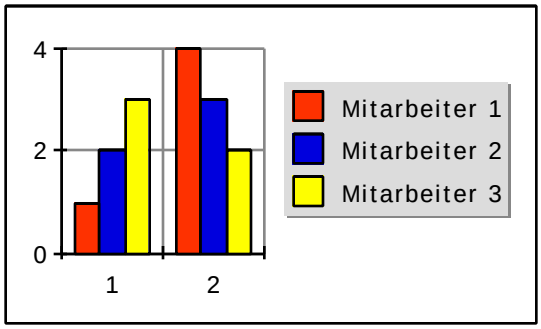
**LegendBackground(füllfarbe;füllmuster;rahmenbreite;
rahmenfarbe;rahmenmuster;schattenabstand;
schattenfarbe;schattenmuster)**

Durch die Funktion LegendBackground() kann der Legendenhintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendBackground()
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendBackground(lightGray;;0;;;1)
CloseDrawing()
```



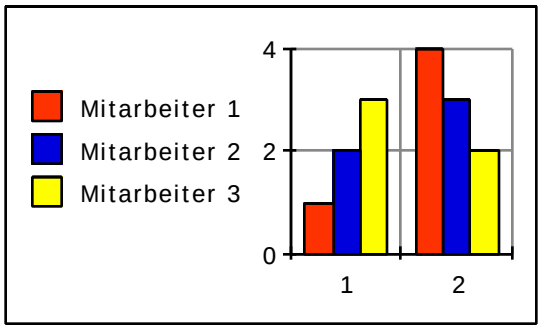
LegendOptions(platzierung;innerhalbDiagramm;hVersatz;vVersatz;zeilenanzahl;symboltyp;symbolbreite;symbolhöhe;symbolabstand;zeilenabstand;spaltenabstand)

Die Funktion `LegendOptions()` stellt zahlreiche Möglichkeiten zur Platzierung und Feinabstimmung der Legende zur Verfügung. Standardmäßig wird die Legende an der rechten Seite des Diagramms platziert. Das erste Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen:

Konstante	*Wert*	*Anmerkung*
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts (default)
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

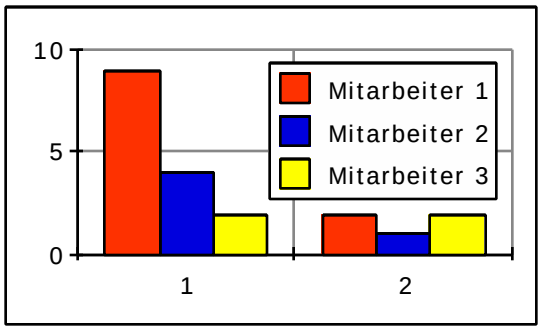
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendOptions(centerLeft)
CloseDrawing()
```



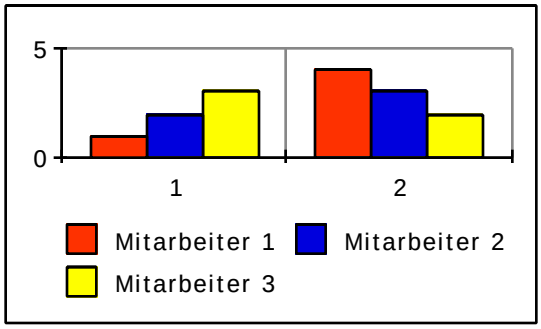
Zusätzlich besteht die Möglichkeit mittels *platzierung=0* die Legende relativ zum Koordinatenursprung zu platzieren. Der Koordinatenursprung befindet sich in der linken oberen Ecke der Zeichnung bzw. bei Vorhandensein eines Views, in der linken oberen Ecke des Views.
Das 2. Argument *innerhalbDiagramm* erlaubt die Positionierung der Legende innerhalb der Diagrammfläche. Optional kann noch durch *hVersatz* und *vVersatz* die Lage der Legende feinabgestimmt werden. Positive Versatzwerte schieben die Legende nach rechts unten, negative Werte nach links oben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 2; 4 1; 2 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendBackground()
     LegendOptions(topRight;on;3;-3)
CloseDrawing()
```



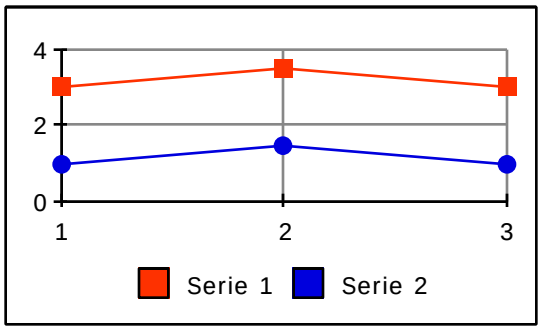
Das Argument *zeilenanzahl* dient zur Kontrolle der Anordnung der Legendentexte. Zum Beispiel, bei *zeilenanzahl=1* werden alle Legendentexte in einer einzigen Zeile aneinander gereiht, bei *zeilenanzahl=2* werden die Legendtexte zweireihig angeordnet, bei *zeilenanzahl=3* in drei Zeilen, usw. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendOptions(bottomCenter;;;;2)
CloseDrawing()
```

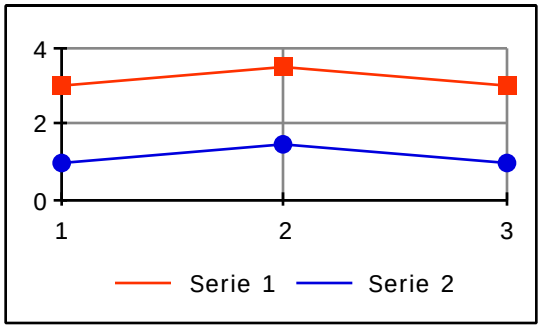


Das Argument *symboltyp* dient zum Festlegen des Legendensymbols. Als Legendensymbol kann ein Rechteck, eine Linie oder ein Diagrammsymbol verwendet werden — auch Kombinationen sind möglich. Wird kein Symboltyp vorgegeben, wird automatisch ein passender Typ gewählt, zum Beispiel bei Balken- und Kuchendiagrammen ein Quadrat, bei Liniendiagrammen mit Symbolen, eine Linie mit einem Symbol überlagert.
Beispiele:

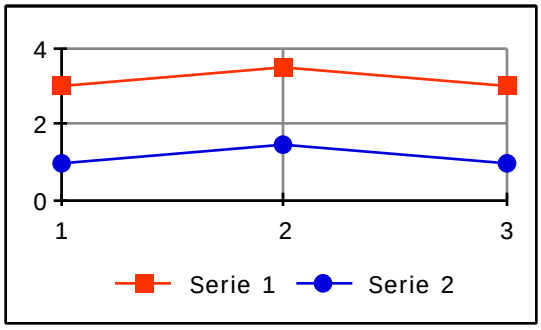
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 3.5 3; 1 1.5 1)
     LineChart(symbol)
     SymbolStyle(1;square;6)
     SymbolStyle(2;bullet;6)
     LegendOptions(bottomCenter;;;;1;rect)
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 3.5 3; 1 1.5 1)
     LineChart(symbol)
     SymbolStyle(1;square;6)
     SymbolStyle(2;bullet;6)
     LegendOptions(bottomCenter;;;;1;line)
CloseDrawing()
```

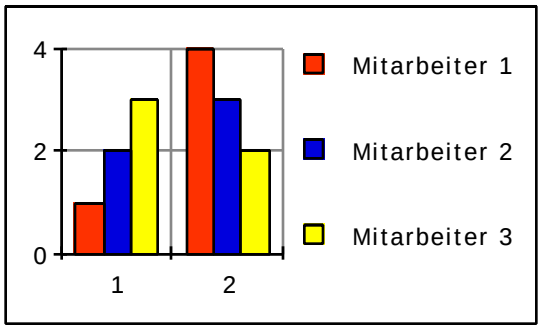



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 3.5 3; 1 1.5 1)
     LineChart(symbol)
     SymbolStyle(1;square;6)
     SymbolStyle(2;bullet;6)
     LegendOptions(bottomCenter;;;;1;line+symbol)
CloseDrawing()
```



Die Argumente *symbolhöhe*, *symbolbreite*, *symbolabstand*, *zeilenabstand* und *spaltenabstand* kontrollieren die Größe des Legendensymbols, den Abstand zwischen Legendensymbol und Legendentext, sowie den Abstand zwischen den Legendentexten. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(1 4; 2 3; 3 2)
     BarChart()
     LegendTexts("Mitarbeiter 1";
                 "Mitarbeiter 2";
                 "Mitarbeiter 3")
     LegendOptions(;;;;;;8;8;10;20)
CloseDrawing()
```



Titel

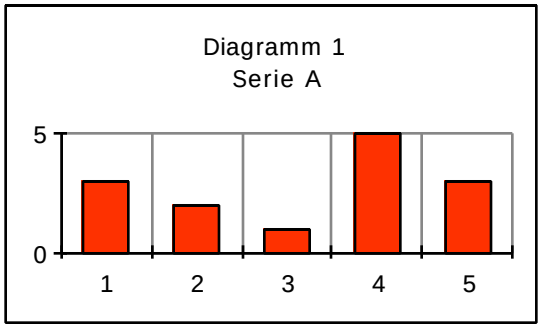
Zur Titelgestaltung stehen insgesamt fünf Funktionen zur Verfügung. Diese ermöglichen:

- das Definieren eines Titeltexts inkl. Untertitel
- getrennte Schriftstile für Titel und Untertitel
- die Gestaltung des Hintergrundes
- ein flexibles Platzieren des Titels

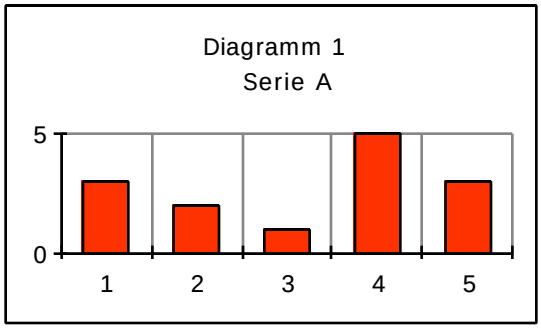
TitleText(titel;untertitel)

Durch die Funktion `TitleText()` wird der Text des Titels festgelegt; optional besteht die Möglichkeit als 2. Argument einen Untertitel zu definieren. Durch Einfügen eines Zeilenumbruchs "\n" können auch mehrzeilige Texte ausgegeben werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1\nSerie A")
CloseDrawing()
```



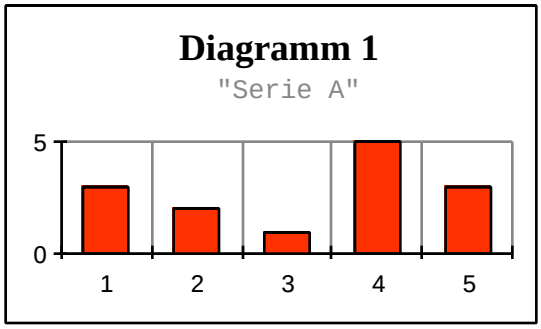

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1";"Serie A")
CloseDrawing()
```



TitleStyle(schrift;größe;stil;farbe;ausrichtung)
TitleSubStyle(schrift;größe;stil;farbe;ausrichtung)

Durch die Funktion `TitleStyle()` bzw. `TitleSubStyle()` wird der Schriftstil des Titels bzw. Untertitels festgelegt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1";"\"Serie A\"")
     TitleStyle("Times";14;bold)
     TitleSubStyle("Courier";10;plain;gray)
CloseDrawing()
```

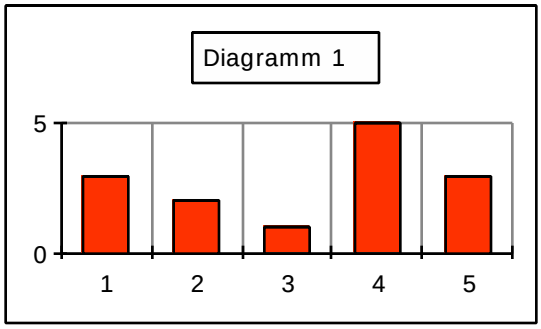


TitleBackground(füllfarbe;füllmuster;rahmenbreite;
rahmenfarbe;rahmenmuster;schattenabstand;
schattenfarbe;schattenmuster)

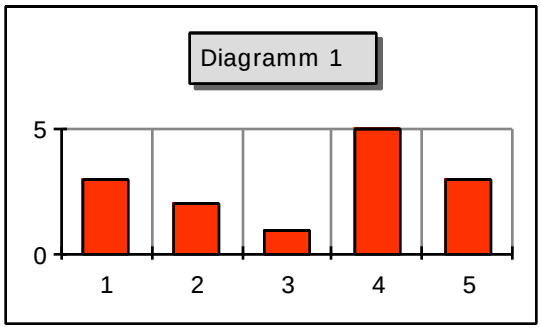
Durch die Funktion `TitleBackground()` kann der Titelhintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1")
     // weißer Hintergrund mit einem
     // 1 Pixel breiten, schwarzen Rand.
     TitleBackground()
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1")
     // hellgrauer Hintergrund mit einem 1 Pixel
     // breiten, schwarzen Rand und einem 2 Pixel
     // breiten, dunkelgrauen Schatten
     TitleBackground(lightGray;;1;;;2;darkGray)
CloseDrawing()
```



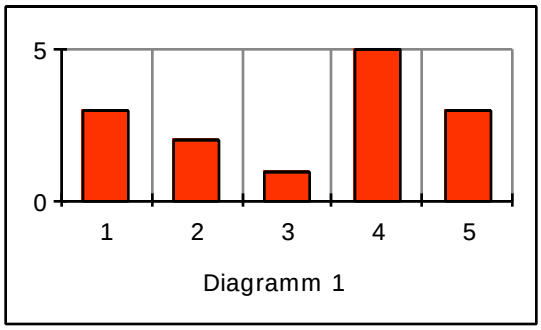
TitleOptions(platzierung;innerhalbDiagramm;hVersatz; vVersatz;vUntertitelversatz;titelausrichtung)

Die Funktion `TitleOptions()` dient zur Platzierung und Ausrichtung des Titels. Standardmäßig wird der Titel in der Mitte oberhalb des Diagramms platziert. Das 1. Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen:

Konstante	*Wert*	*Anmerkung*
`topLeft`	1	oben links
`topCenter`	2	oben mitte (default)
`topRight`	3	oben rechts
`centerLeft`	4	mitte links
`centerCenter`	5	mitte mitte
`centerRight`	6	mitte rechts
`bottomLeft`	7	unten links
`bottomCenter`	8	unten mitte
`bottomRight`	9	unten rechts

Beispiel:

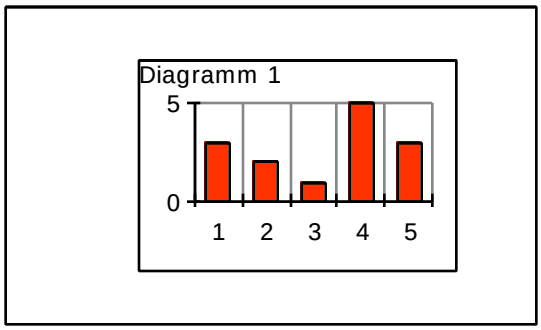
```
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    ChartData(3 2 1 5 3)
    BarChart()
    TitleText("Diagramm 1")
    TitleOptions(bottomCenter)
CloseDrawing()
```



Zusätzlich besteht die Möglichkeit mittels *platzierung=0* den Titel relativ zum Koordinatenursprung zu platzieren. Der Koordinatenursprung befindet sich in der linken oberen Ecke der Zeichnung oder, falls das Diagramm innerhalb eines Views platziert wird, in der linken oberen Ecke des Views.

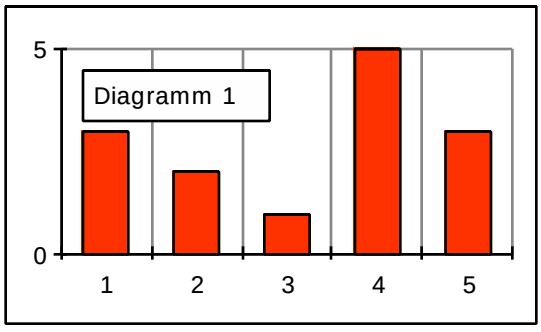
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     OpenView(50;20;120;80)
          AddFrame(0;0;120;80)
          ChartData(3 2 1 5 3)
          BarChart()
          TitleText("Diagramm 1")
          TitleOptions(0)
     CloseView()
CloseDrawing()
```



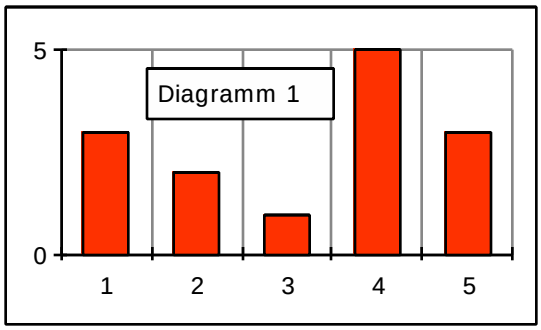
Das 2. Argument *innerhalbDiagramm* erlaubt die Positionierung des Titels innerhalb der Diagrammfläche. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1")
     TitleBackground()
     TitleOptions(topLeft;on)
CloseDrawing()
```



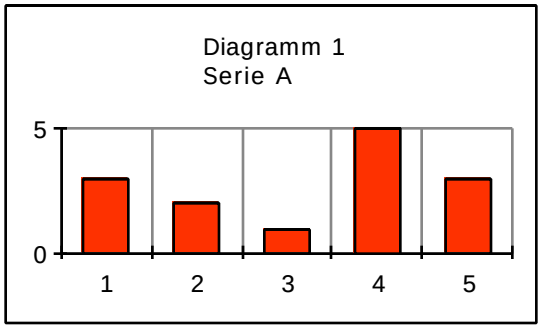
Optional kann durch *hVersatz* und *vVersatz* die Lage des Titels feinabgestimmt werden. Positive Versatzwerte schieben den Titel nach rechts unten, negative Werte nach links oben. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1")
     TitleBackground()
     TitleOptions(topLeft;on;24;-1)
CloseDrawing()
```



Der Abstand und die Ausrichtung zwischen Titel und Untertitel kann durch *vUntertitelversatz* und *titelausrichtung* kontrolliert werden. Ein positiver Versatzwert vergrößert den Abstand zwischen Titel und Untertitel, ein negativer Wert verkleinert den Abstand. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
     TitleText("Diagramm 1";"Serie A")
     TitleOptions(topCenter;off;0;0;-2;left)
CloseDrawing()
```



Hilfslinien

Punkt-, Linien- und Blasendiagramme können optional durch Hilfslinien ergänzt werden. Dazu stehen insgesamt vier Funktionen zur Verfügung. Diese dienen:

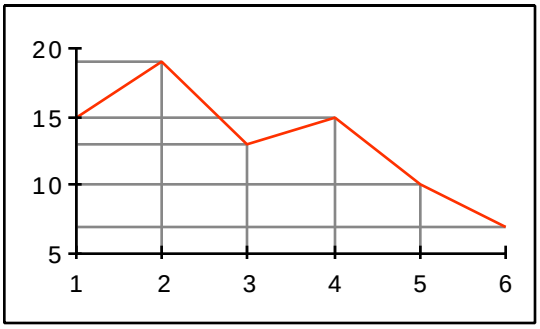
- zur Gestaltung der Hilfslinien
- zum Festlegen eines Referenzpunktes
- zum Festlegen einer Referenzlinie
- zum Festlegen von Referenzserien

Bei allen vier Funktionen wird als erstes Argument der Serienindex festgelegt. Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle Datenserien.

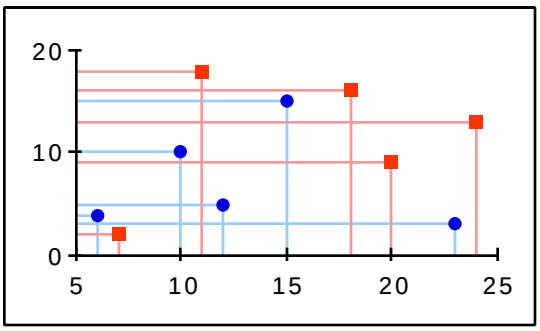
DropLineStyle(serienindex;achsenindex;strichstärke; farbe;muster)

Das 2. Argument *achsenindex* legt die Richtung der Hilfslinien fest. Bei *achsenindex=x* werden die Hilfslinien auf die x-Achse bezogen, bei *achsenindex=y* auf die y-Achse. Wird kein Achsenindex vorgegeben bzw. ist *achsenindex=all*, werden die Hilfslinien sowohl in Richtung der x-Achse als auch in Richtung der y-Achse gezeichnet. Die Strichstärke, die Farbe und das Muster der Hilfslinien können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Hilfslinien als graue, 1 Pixel breite Linien dargestellt. Beispiele:

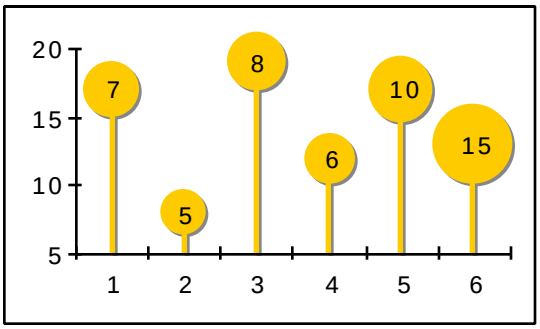
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 19 13 15 10 7)
     LineChart()
     DropLineStyle()
     AxisOptions(all;front) // Achsen vor Hilfslinien
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 18 20 24; // x-Werte 1.Serie
                2 18 16  9 13; // y-Werte 1.Serie
               23 10 15 12  6; // x-Werte 2.Serie
                3 10 15  5  4) // y-Werte 2.Serie
     ScatterChart2D()
     SymbolStyle(1;square;4;;red)
     SymbolStyle(2;bullet;4;;blue)
     DropLineStyle(1;all;1;lightRed)
     DropLineStyle(2;all;1;lightBlue)
     AxisOptions(all;front) // Achsen vor Hilfslinien
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(17 8 19 12 17 13; // y-Positionen
               7 5  8  6 10 15) // Durchmesser
     BubbleChart(label+shadow;on)
     FillStyle(1;darkYellow)
     BorderStyle(1;none)
     ShadowStyle(1;1;gray)
     DropLineStyle(1;x;2;darkYellow)
     AxisOptions(all;front) // Achsen vor Hilfslinien
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

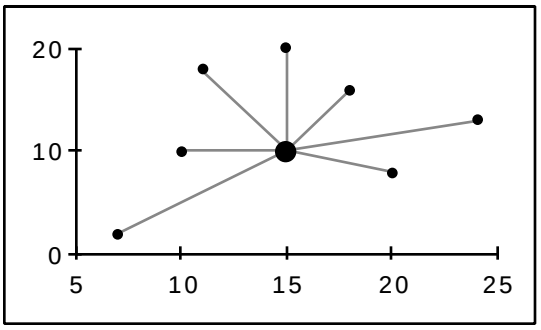


DropLineReferencePoint(serienindex;xPunkt;yPunkt;
symboltyp;symbolgröße;strichstärke;
symbolfarbe;symbolmuster)

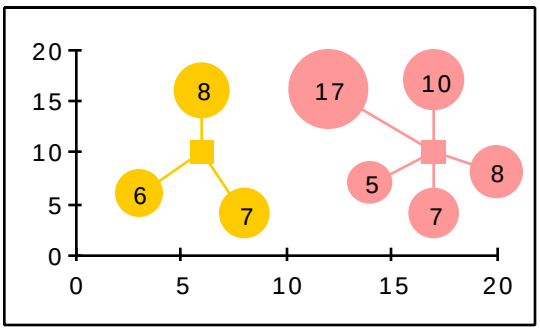
Die Funktion `DropLineReferencePoint()` dient zur Festlegung eines Referenzpunktes, von welchem alle Hilfslinien strahlenförmig ausgehen. In diesem Fall werden als Argument *achsenindex* in der Funktion `DropLineStyle()` alle Indizes größer 1 ignoriert, d.h. es wird nur *achsenindex=x* oder *achsenindex=1* akzeptiert. Die Argumente *xPunkt* und *yPunkt* legen die Position des Referenzpunktes fest. Die Gestaltung des Referenzpunktes erfolgt durch die Argumente *symboltyp*, *symbolgröße*, *strichstärke*, *symbolfarbe* und *symbolmuster*. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Standardmäßig wird der Referenzpunkt durch einen schwarzen, kreisförmigen Punkt (*symboltyp=bullet*) mit 9 Pixel Durchmesser dargestellt.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 18 20 24 10 15; // x-Werte
                2 18 16  8 13 10 20) // y-Werte
     ScatterChart2D()
     SymbolStyle(1;bullet;3;;black)

     DropLineReferencePoint(1;15;10)
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```



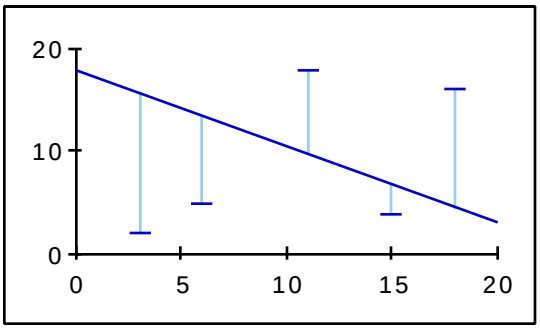

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(17 14 20 12 17; // x-Positionen 1.Serie
               4  7  8 16 17; // y-Positionen 1.Serie
               7  5  8 17 10; // Durchmesser  1.Serie
               8  3  6;       // x-Positionen 2.Serie
               4  6 16;       // y-Positionen 2.Serie
               7  6  8)       // Durchmesser  2.Serie
     BubbleChart2D(label)
     BorderStyle(all;none)
     FillStyle(1;lightRed)
     FillStyle(2;darkYellow)
     DropLineStyle(1;;1;lightRed)
     DropLineStyle(2;;1;darkYellow)
     DropLineReferencePoint(1;17;10;square;8;;lightRed)
     DropLineReferencePoint(2;6;10;square;8;;darkYellow)
     GridLocation(xy;none) // kein Raster
CloseDrawing()
```



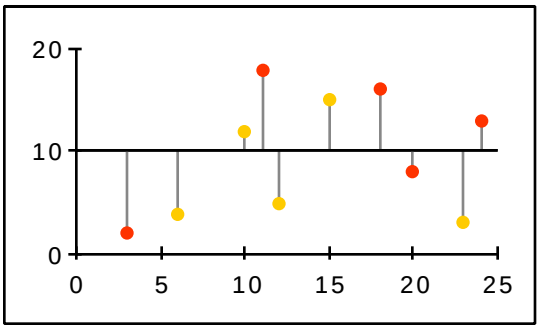
**DropLineReferenceLine(serienindex;xStart;yStart;
xEnde;yEnde;strichstärke;farbe;
muster)**

Die Funktion `DropLineReferenceLine()` dient zur Festlegung einer Referenzlinie. Die Argumente *xStart*, *yStart* und *xEnde*, *yEnde* definieren die Referenzlinie. Die Gestaltung der Referenzlinie erfolgt durch die Argumente *strichstärke*, *farbe* und *muster*. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 11 18 15 6; // x-Werte
               2 18 16  4 5) // y-Werte
     ScatterChart2D()
     SymbolStyle(1;hBar;8;;blue)
     DropLineStyle(1;x;1;lightBlue)
     DropLineReferenceLine(1;0;18;20;3;1;blue)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 3 11 18 20 24; // x-Werte 1.Serie
                2 18 16  8 13; // y-Werte 1.Serie
               23 10 15 12  6; // x-Werte 2.Serie
                3 12 15  5  4) // y-Werte 2.Serie
     ScatterChart2D()
     SymbolStyle(1;bullet;4;;red)
     SymbolStyle(2;bullet;4;;darkYellow)
     DropLineReferenceLine(all;0;10;25;10)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

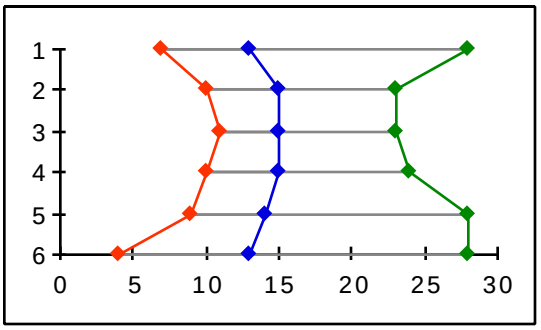


DropLineReferenceSeries(serienindex;
referenzSerienindex1;
referenzSerienindex2...)

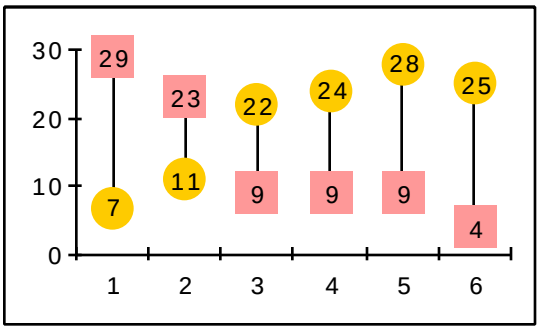
Datenpunkte unterschiedlicher Serien können unter Verwendung der Funktion `DropLineReferenceSeries()` durch Hilfslinien verbunden werden. Die "Bezugsserie" wird durch das 1. Argument *serienindex* festgelegt.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 10 11 10  9  4; // 1.Serie
               13 15 15 15 14 13; // 2.Serie
               28 23 23 24 28 28) // 3.Serie
     LineChart(horizontal+symbol)
     LineStyle(all;poly;1)
     LineStyle(3;poly;1;green)
     SymbolStyle(all;diamond;4;1)
     SymbolStyle(3;diamond;4;1;green)
     DropLineReferenceSeries(all;1)
     ScalingOptions(y;on)   // y-Skala von oben nach unten
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 22 24 28 25; // 1.Serie
               29 23  9  9  9  4) // 2.Serie
     ScatterChart(label;on)
     SymbolStyle(1;bullet;15;;darkYellow)
     SymbolStyle(2;square;15;;lightRed)
     LabelOptions(all;centerCenter)
     DropLineStyle(all;x;1;black)
     DropLineReferenceSeries(1;2)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```


30
20
10
0
29
7
23
11
22
9
24
9
28
9
25
4
1
2
3
4
5
6

Gleitende Durchschnitte

Punkt- und Liniendiagramme können optional durch gleitende Durchschnitte ergänzt werden. Dazu stehen insgesamt drei Funktionen zur Verfügung. Diese ermöglichen:

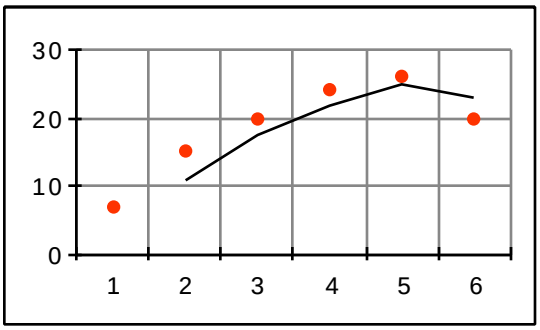
- die Festlegung unterschiedlicher Berechnungsmethoden
- die Gestaltung der Durchschnittslinien
- die Feinabstimmung mittels zahlreicher Optionen

Bei allen drei Funktionen werden als erstes Argument der Serienindex und als zweites Argument die Intervallanzahl festgelegt. Die Intervallanzahl bestimmt die Anzahl der zur Berechnung des Durchschnitts herangezogenen Datenpunkte. Soll zum Beispiel ein 50-Tage-Durchschnitt berechnet werden, so ist *intervallanzahl=50* zu setzen. Wird keine Intervallanzahl vorgegeben, so werden standardmäßig die Durchschnittswerte aus zwei Datenpunkten errechnet (*intervallanzahl=2*). Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle vorhandenen Datenserien.

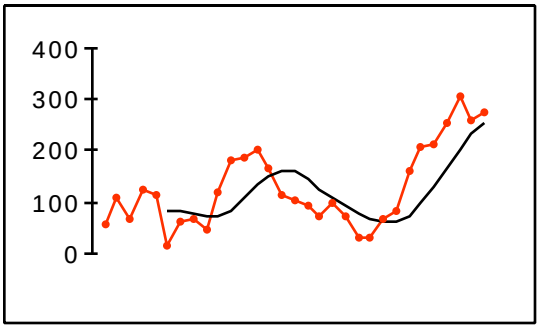
MovingAverage(serienindex;intervallanzahl; wichtungsfaktoren)

Das erste Argument *serienindex* in der Funktion `MovingAverage()` legt fest, für welche Datenserie der gleitende Durchschnitt zu berechnen ist. Das zweite Argument *intervallanzahl,* definiert die Anzahl der Datenpunkte zur Berechnung der Durchschnittswerte. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 24 26 20)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage()
CloseDrawing()
```



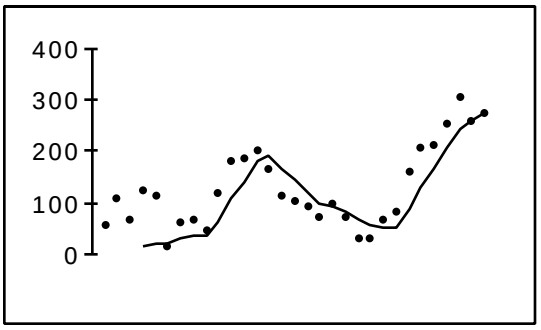

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274)
     LineChart(symbol)
     SymbolStyle(1;bullet;2)
     MovingAverage(1;6)     // 6 Pkte zur Mittelwertbildg.
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



Optional kann als drittes Argument eine Liste von Wichtungsfaktoren angeführt werden. Dabei wird der 1. Datenpunkt mit dem 1. Wichtungsfaktor multipliziert, der 2. Datenpunkt mit dem 2. Wichtungsfaktor, usw. Ein Wichtungsfaktor kleiner 1 verringert den Einfluss eines Datenpunktes zur Mittelwertbildung, ein Wichtungsfaktor größer 1 vergrößert den Einfluss. Ist die Anzahl der Wichtungsfaktoren kleiner als die Anzahl der Datenpunkte, so werden die fehlenden Faktoren durch den neutralen Wert 1 ergänzt. Die Wichtungsfaktoren sind analog wie in der Funktion `ChartData()` getrennt durch Leerzeichen einzugeben.

Beispiel:

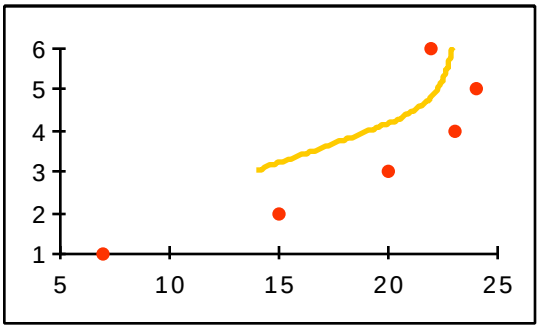
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274)
     ScatterChart()
     SymbolStyle(1;bullet;2;1;black)
     MovingAverage(1;4;.1 .1 .1 .3 .3 .5 .7 .8 .9 1 1.2)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



MovingAverageLineStyle(serienindex;intervallanzahl; darstellung;strichstärke;farbe;muster)

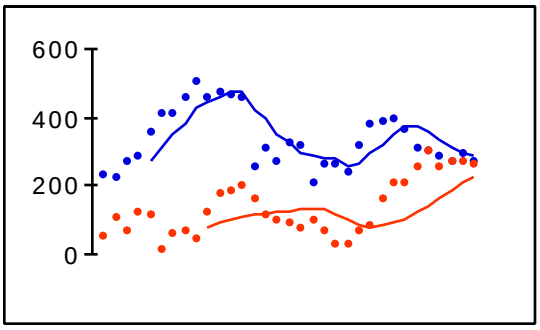
Mit der Funktion `MovingAverageLineStyle()` kann das Aussehen der gleitenden Durchschnittslinien kontrolliert werden. Durch das 3. Argument *darstellung* kann der Kurvenverlauf — geglättet, polygonartig oder stufenförmig — gesteuert werden; Details dazu finden sich unter der Funktion `LineStyle()` im Abschnitt *Stile*. Die Strichstärke, die Farbe und das Muster der Durchschnittslinien können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Durchschnittslinien als schwarze, 1 Pixel breite Linien dargestellt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 23 24 22)
     ScatterChart(horizontal)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;3)
     MovingAverageLineStyle(1;3;smooth;2;darkYellow)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

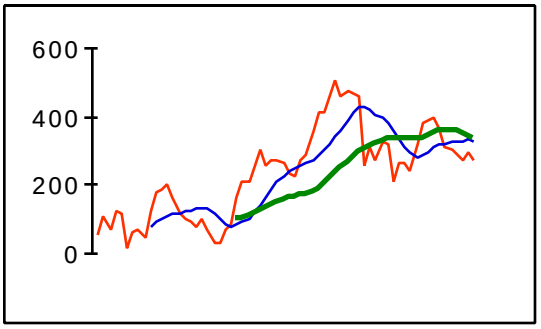


Durch mehrfaches Aufrufen der Funktion `MovingAverage()` können gleichzeitig mehrere gleitende Durchschnitte sowohl für unterschiedliche Datenserien als auch für eine unterschiedliche Anzahl an Intervallen dargestellt werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263;
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     ScatterChart()
     SymbolStyle(all;bullet;2)
     MovingAverage(1;10)
     MovingAverage(2;5)
     MovingAverageLineStyle(1;10;poly;1;red)
     MovingAverageLineStyle(2;5;poly;1;blue)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     LineChart()
     MovingAverage(1;10)
     MovingAverage(1;25)
     MovingAverageLineStyle(1;10;poly;1;blue)
     MovingAverageLineStyle(1;25;poly;2;green)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

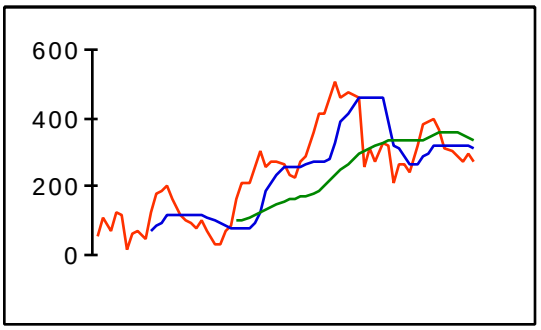


MovingAverageOptions(serienindex;intervallanzahl;
berechnungsmethode;ausrichtung;extrapolieren;
hVersatz;vVersatz;istRelHVersatz;
istRelVVersatz)

Durch die Funktion `MovingAverageOptions()` kann die Berechnung und Darstellung gleitender Durchnitte vielfältig variiert werden. So kann mit dem 3. Argument *berechnungsmethode* unter drei verschiedenen Methoden zur Berechnung der Durchschnittswerte gewählt werden. Standardmäßig wird die Durchschnittslinie aus den arithmetischen Mittelwerten errechnet (*berechnungsmethode=average*). Anstelle der Mittelwerte ist es auch möglich, die Mediane zu verwenden (*berechnungsmethode=median*).

Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     LineChart()
     MovingAverage(1;10)
     MovingAverage(1;25)
     MovingAverageLineStyle(1;10;poly;1;blue)
     MovingAverageLineStyle(1;25;poly;1;green)
     MovingAverageOptions(1;10;median)
     MovingAverageOptions(1;25;average)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



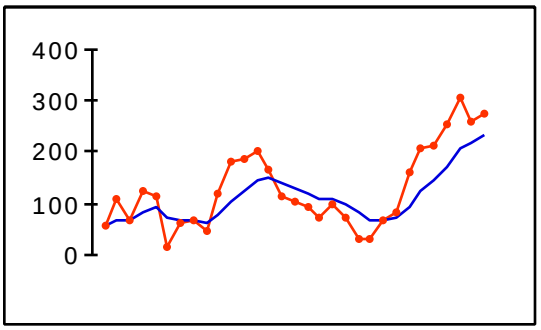
Die dritte Möglichkeit besteht in der Berechnung eines exponentiell gewichteten Durchschnitts (*berechnungsmethode=exponential*) nach der Formel:

```
w[i+1] = w[i] + a*(y[i+1]-w[i])
   mit:  a.....Glättungskonstante zwischen 0 und 1
         y[]...Originalwert
         w[]...geglätteter Wert
```

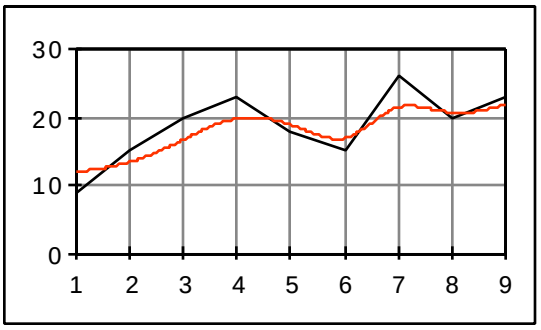
Die dazu benötigte Glättungskonstante wird als drittes Argument in der Funktion `MovingAverage()` anstelle der Wichtungsfaktoren übergeben. Zusätzlich kann zur Glättungskonstante noch optional ein Startwert hinzugefügt werden. Im Falle einer exponentiellen Glättung werden die Argumente *ausrichtung* und *extrapolieren* ignoriert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116  13  63  67  45 121 181
               189 201 166 115 105  91  75  99  72  33  29
                69  85 162 210 211 256 304 258 274)
     LineChart(symbol)
     SymbolStyle(all;bullet;2)
     MovingAverage(1;;0,25) // 0,25 = Glättungskonst.
     MovingAverageLineStyle(1;2;poly;1;blue)
     MovingAverageOptions(1;;exponential)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

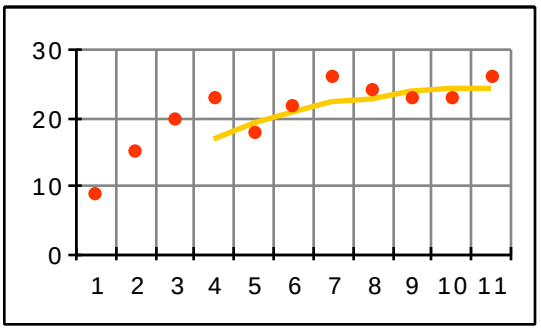



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 15 26 20 23)
     LineChart()
     LineStyle(1;poly;1;black)

     // 0.5...Glättungskonstante
     // 15 ...Startwert
     MovingAverage(1;;0.5 15)
     MovingAverageLineStyle(1;;smooth;1;red)
     MovingAverageOptions(1;;exponential)
     GridFrame()
CloseDrawing()
```



Durch das 4. Argument *ausrichtung* kann die Position der Durchschnitttslinie kontrolliert werden. Durchschnittslinien — ausgenommen bei exponentieller Glättung — sind immer "etwas kürzer" als die Originalkurve, da die Punkteanzahl zur Darstellung der Durchschnittslinien stets um den Wert *intervallanzahl-1* geringer ist als die Punkteanzahl der Originalkurve. D.h., zum Beispiel eine 50-Tage-Durchschnittslinie besitzt um 49 Punkte weniger als die Ausgangskurve. Zur Ausrichtung der kürzeren Durchschnittslinien stehen vier Konstanten zur Verfügung. Standardmäßig wird die Durchschnittslinie nach dem letzten Datenpunkt ausgerichtet (*ausrichtung=backward*). Beispiel:

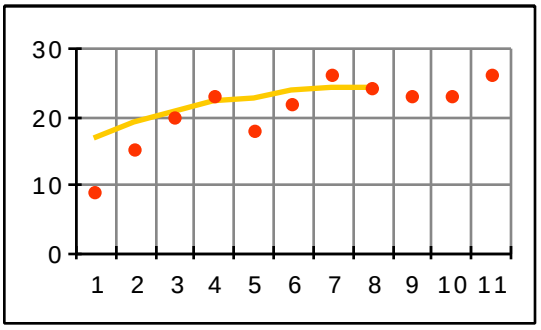
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;4)
     MovingAverageLineStyle(1;4;poly;2;darkYellow)
     MovingAverageOptions(1;4;average;backward)
CloseDrawing()
```



Die Durchschnittslinie wird bündig mit dem ersten Datenpunkt ausgerichtet, falls *ausrichtung=forward* gesetzt wird.

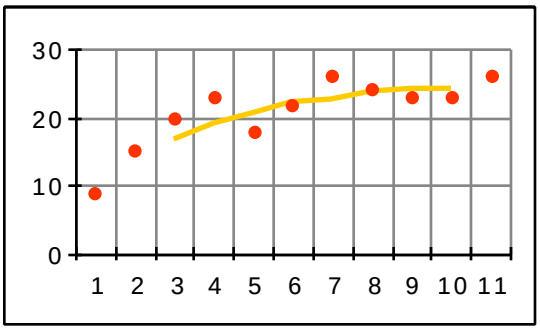
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;4)
     MovingAverageLineStyle(1;4;poly;2;darkYellow)
     MovingAverageOptions(1;4;average;forward)
CloseDrawing()
```

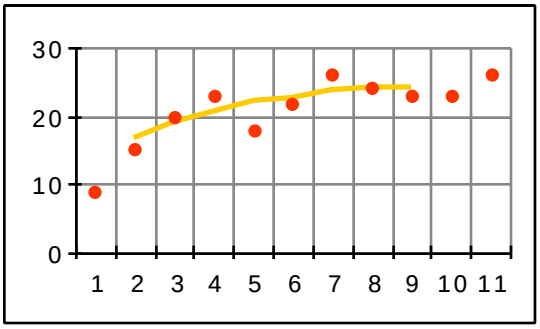


Bei *ausrichtung=centeredBackward* bzw. *ausrichtung=centeredForward* wird die Durchschnittslinie "mittig" ausgerichtet. Das heißt, zum Beispiel bei einer Intervallanzahl von 50 wird bei *ausrichtung=centeredBackward* die Durchschnittslinie um 25 Punkte eingerückt, bei *ausrichtung=centered-Forward* um 24 Punkte. Falls die Anzahl der fehlenden Punkte geradzahlig ist, liefern die beiden Konstanten *centeredForward* und *centeredBackward* das gleiche Ergebnis. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;4)
     MovingAverageLineStyle(1;4;poly;2;darkYellow)
     MovingAverageOptions(1;4;average;centeredBackward)
CloseDrawing()
```

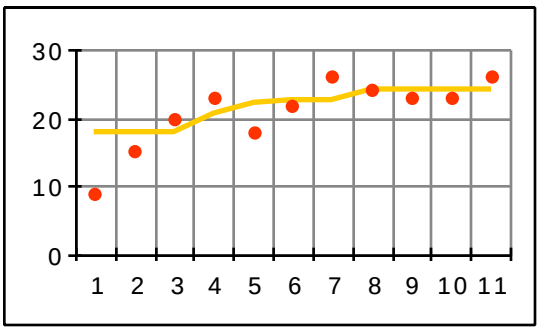



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;4)
     MovingAverageLineStyle(1;4;poly;2;darkYellow)
     MovingAverageOptions(1;4;average;centeredForward)
CloseDrawing()
```



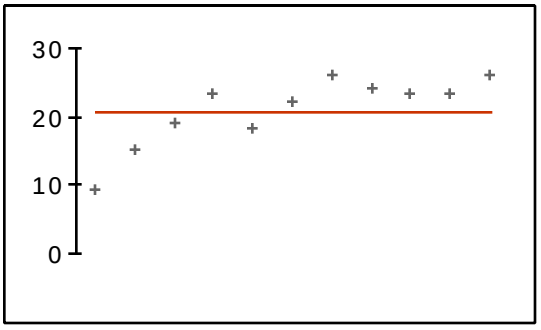
Durch Aktivieren des Arguments *extrapolieren=on* ist es möglich, die Durchschnittslinie über den gesamten Datenbereich zu verlängern. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 20 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;bullet;4;1;red)
     MovingAverage(1;6)
     MovingAverageLineStyle(1;6;poly;2;darkYellow)
     MovingAverageOptions(1;6;average;centeredForward;on)
CloseDrawing()
```



Als vorteilhaft erweist sich *extrapolieren=on* bei der Darstellung des Gesamtdurchschnitts. Der Gesamtdurchschnitt ergibt sich allgemein für eine aus *n* Werten bestehende Datenserie, indem *intervallanzahl* gleich *n* gesetzt wird. Beispiel:

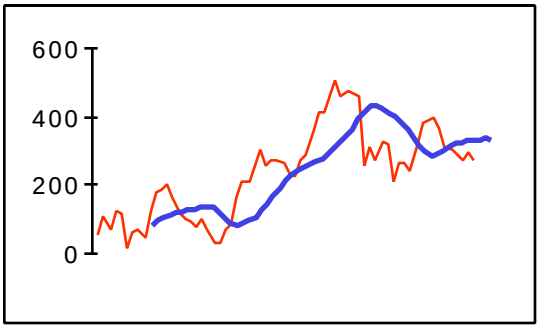
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(9 15 19 23 18 22 26 24 23 23 26)
     ScatterChart(;on)
     SymbolStyle(all;plus;5;1;darkGray)
     MovingAverage(1;11)
     MovingAverageLineStyle(1;11;poly;1;darkRed)
     MovingAverageOptions(1;11;average;;on)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



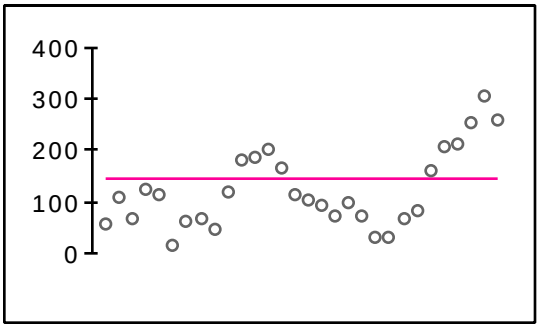
Gleitende Durchschnitte können optional durch Vorgabe eines horizontalen Versatzwertes (*hVersatz*) und/oder eines vertikalen Versatzwertes (*vVersatz*) verschoben dargestellt werden. Dabei können Versatzwerte entweder als absolute Größen (*istRelHVersatz=off* bzw. *istRelVVersatz=off*) vorgegeben werden (default) oder prozentuell bezogen auf die Durchschnittswerte (*istRelHVersatz=on* bzw. *istRelVVersatz=on*).

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108 70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     LineChart()
     MovingAverage(1;10)
     MovingAverageLineStyle(1;10;poly;2;blue;darkGray)
     // um 5% nach rechts verschieben
     MovingAverageOptions(1;10;average;;;5;;on)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108 70 125 116 13 63 67
                45 121 181 189 201 166 115
               105 91 75 99 72 33 29 69 85
               162 210 211 256 304 258)
     ScatterChart()
     SymbolStyle(all;circle;4;1;darkGray)
     MovingAverage(1;30)
     MovingAverageLineStyle(1;30;poly;1;purple)
     // um 20 Einheiten nach oben verschieben
     MovingAverageOptions(1;30;average;;on;;20)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```


400
300
200
100
0

Kurvenanpassung

Zur Kurvenanpassung bei Punkt- und Liniendiagrammen stehen drei Funktionen zur Verfügung. Diese ermöglichen:

- die Festlegung unterschiedlicher Kurvenfunktionen
- die Gestaltung der Kurven
- die Feinabstimmung mittels zahlreicher Optionen

Bei allen drei Funktionen werden als erstes Argument der Serienindex und als zweites Argument der Kurvenfunktionstyp festgelegt. Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle vorhandenen Datenserien. Folgende Kurvenfunktionen stehen zur Verfügung:

- Polynomfunktionen bis max. 10. Ordnung. (*typ=1...10*)
  f(x) = c0 + c1*x + c2*x^2 + c3*x^3 + ... + cN*x^N
  Für den häufig benötigten Fall eines linearen Polynoms (Gerade) f(x) = c0 + c1*x steht zusätzlich die Konstante *linear* zur Verfügung (*typ=linear*).
- Potenzfunktionen (*typ=pow*):
  f(x) = c0 * x^c1
- Exponentialfunktion (*typ=exp*):
  f(x) = c0 * exp(c1*x)
- Logarithmusfunktion (*typ=log*):
  f(x) = c0 + c1*ln(x)

Die Berechnung der Kurvenkoeffizienten *c0...c10* erfolgt nach der Methode der kleinsten Fehlerquadrate.
Mit der externen Funktion `xmCH-GetCFValue()` können der Korrelationskoeffizient *r^2* und die Kurvenkoeffizienten *c0* bis max. *c10* in eine FileMaker Pro Datei importiert werden. Die Funktion `xmCH-GetCFValue()` besitzt maximal fünf Argumente, welche durch einen vertikalen Balken "|" getrennt einzugeben sind:

```
xmCH-GetCFValue(Diagrammindex|
                Serienindex|
                Kurvenfunktionsindex|
                Werteindex|
                Formatanweisung)
```

- Diagrammindex: (erforderlich)
  Da innerhalb einer Zeichnung mehrere Diagramme gleichzeitig dargestellt werden können, ist es notwendig, Diagramme durch einen Index zu spezifizieren. Der Index entspricht der Reihen-

folge der Diagrammdefinitionen innerhalb von `OpenDrawing()` und `CloseDrawing()`. Besteht die Zeichnung nur aus einem einzigen Diagramm, so ist der Diagrammindex gleich 1.

- Serienindex: (erforderlich)
  Mit dem Serienindex wird festgelegt, auf welche Serie sich die Abfrage bezieht. Dies ist notwendig, da Kurvenanpassungen für mehrere Serien gleichzeitig durchgeführt werden können.
- Kurvenfunktionsindex: (erforderlich)
  Mit dem Kurvenfunktionsindex wird festgelegt, auf welche Kurvenfunktion zugegriffen werden soll. Dies ist erforderlich, da für eine Datenserie mehrere Kurvenanpassungen gleichzeitig berechnet werden können. Der Kurvenfunktionsindex ist ident dem Argument *typ* und besitzt einen Wert zwischen -4 im Falle einer logarithmische Kurvenanpassung und 10 für ein Polynom 10. Ordnung.
- Werteindex: (erforderlich)
  Der Werteindex erlaubt den Zugriff auf die einzelnen Koeffizienten. Werteindex gleich 1 liefert den Korrelationskoeffizienten *r^2*, Werteindex gleich 2 den Kurvenkoeffizienten *c0*, der Werteindex 3 den Kurvenkoeffizienten *c1*, usw.
- Formatanweisung: (optional)
  Optional kann der von der Funktion *xmCH-GetCFValue()* retournierte Koeffizient durch Vorgabe einer Formatanweisung entsprechend formatiert werden. Wird keine Formatanweisung vorgegeben, wird das Defaultformat "|u|" verwendet. Sämtliche Formatanweisungen inkl. zahlreicher Beispiele finden sich in *xmReferenz*.

Die Arbeitsweise der externen Funktion `xmCH-GetCFValue()` kann anhand der Datei *xmCFDemo.FP3* bzw. *xmCFDemo.FP5* studiert werden. Die Dateien sind frei zugänglich und stehen als kostenloser Download zur Verfügung.

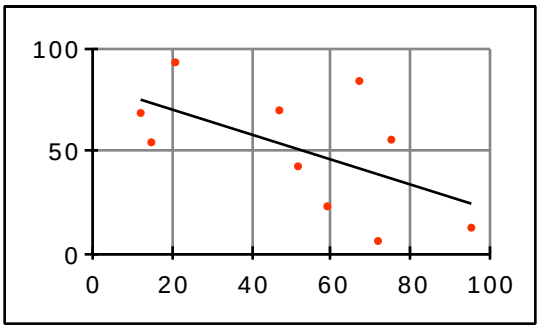
CurveFitting(serienindex;typ)

Das erste Argument *serienindex* in der Funktion `CurveFitting()` legt fest, für welche Datenserie die Kurvenanpassung zu berechnen ist. Das zweite Argument *typ* definiert die Kurvenfunktion. Dazu stehen die folgenden Konstanten zur Verfügung:

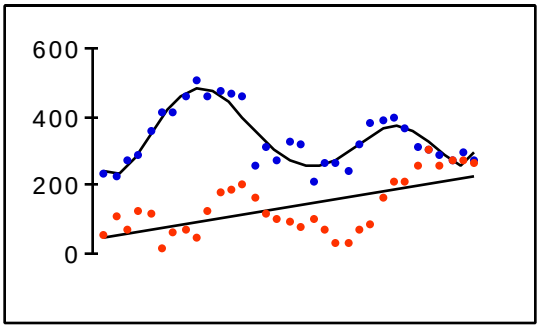
Konstante	*Wert*	*Funktion*
`log`	-4	f(x) = c0 + c1*ln(x)
`exp`	-3	f(x) = c0 * exp(c1*x)
`pow`	-2	f(x) = c0 * x^c1
`linear`	1	f(x) = c0 + c1*x (default)

Wird *typ* nicht vorgegeben, so wird ein Polynom 1. Ordnung (Ausgleichsgerade) berechnet. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(15 59 72 12 47 21 52 75 67 95; // x-Werte
               54 23  7 69 70 94 43 56 85 13) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;bullet;2)
     CurveFitting()
CloseDrawing()
```



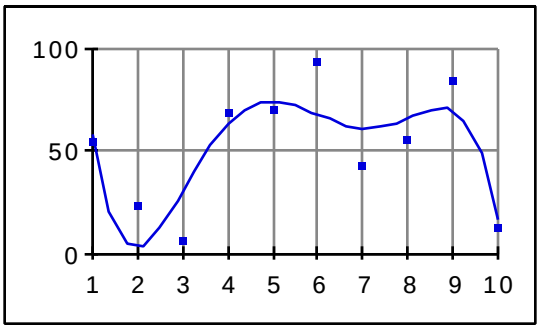

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263;
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     ScatterChart()
     SymbolStyle(all;bullet;2)
     CurveFitting(1;1)
     CurveFitting(2;7)
     AxisOptions(x;none)    // keine x-Achse
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



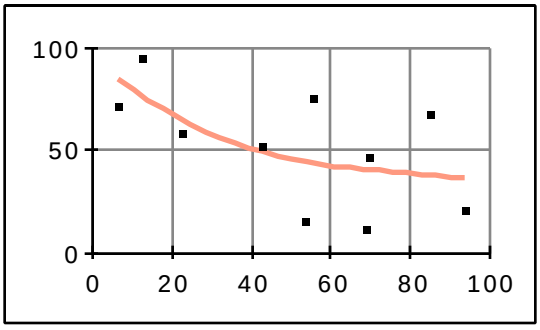
CurveFittingLineStyle(serienindex;typ;strichstärke; farbe;muster)

Mit der Funktion `CurveFittingLineStyle()` kann das Aussehen der Kurven kontrolliert werden. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden die Kurven schwarz und 1 Pixel breit dargestellt. Beispiele:

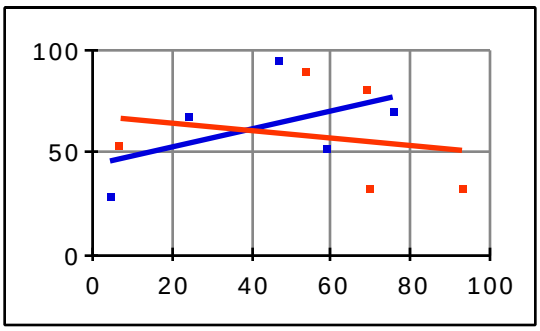
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 23 7 69 70 94 43 56 85 13)
     ScatterChart()
     SymbolStyle(all;square;2;1;blue)
     CurveFitting(1;5)
     CurveFittingLineStyle(1;5;1;blue)
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 23  7 69 70 94 43 56 85 13; // x-Werte
               15 59 72 12 47 21 52 75 67 95) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;square;2;1;black)
     CurveFitting(1;3)
     CurveFittingLineStyle(1;3;2;red;gray)
CloseDrawing()
```



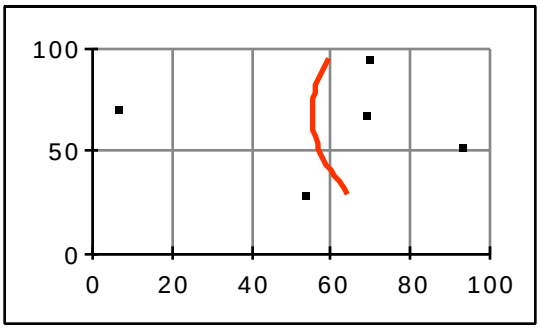

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 93  7 69 70; // x-Werte 1.Serie
               90 33 53 80 33; // y-Werte 1.Serie
                5 59 76 24 47; // x-Werte 2.Serie
               29 52 70 67 95) // y-Werte 2.Serie
     ScatterChart2D()
     SymbolStyle(all;square;2)
     CurveFitting(all;1)
     CurveFittingLineStyle(1;1;2;red)
     CurveFittingLineStyle(2;1;2;blue)
CloseDrawing()
```



**CurveFittingOptions(serienindex;typ;achsenVertauschen;
extrapolieren;nulldurchgang;punkteanzahl)**

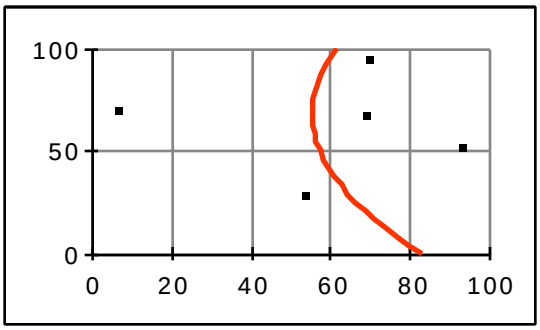
Die Funktion `CurveFittingOptions()` ermöglicht Variationen in der Berechnung und Darstellung der Kurven. Durch Aktivieren des Arguments *achsenVertauschen=on* werden die Koordinaten der zur Kurvenberechnung verwendeten Punkte vertauscht. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 93  7 69 70; // x-Werte
               29 52 70 67 95) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;square;2;1;black)
     CurveFitting(all;2)
     CurveFittingLineStyle(1;2;2;red)
     CurveFittingOptions(all;2;on)
CloseDrawing()
```

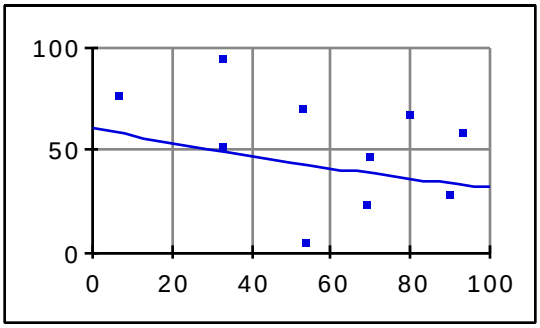


Durch Aktivieren des Arguments *extrapolieren=on* können Kurven bis zum Rand des Diagramms verlängert werden. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 93  7 69 70; // x-Werte
               29 52 70 67 95) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;square;2;1;black)
     CurveFitting(all;2)
     CurveFittingLineStyle(1;2;2;red)
     CurveFittingOptions(all;2;on;on)
CloseDrawing()
```



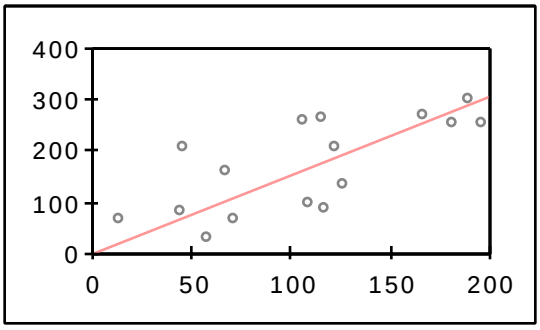

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(54 93  7 69 70 90 33 53 80 33;
               5 59 76 24 47 29 52 70 67 95)
     ScatterChart2D()
     SymbolStyle(all;square;2;1;blue)
     CurveFitting(1;exp)
     CurveFittingLineStyle(1;exp;1;blue)
     CurveFittingOptions(1;exp;;on)
CloseDrawing()
```



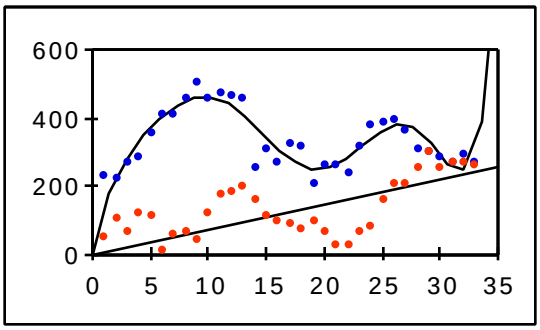
Polynomfunktionen verlaufen durch den Koordinatenursprung falls das Argument *nulldurchgang=on* gesetzt wird. Bei allen anderen Kurvenfunktionen wird das Argument *nulldurchgang* ignoriert.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(57 108 70 125 116 13 43 67 45 121
               181 189 195 166 115 105 91;
               35 99 72 139 90 69 85 162 210 211
               256 304 258 274 270 263)
     ScatterChart2D()
     SymbolStyle(all;circle;3;1;gray)
     CurveFitting(1;linear)
     CurveFittingLineStyle(1;linear;1;lightRed)
     CurveFittingOptions(1;linear;;on;on)
     MajorGridLineWidths(all;all;0) // hide grid
     GridLocation(xy;front)
     GridFrame()
CloseDrawing()
```

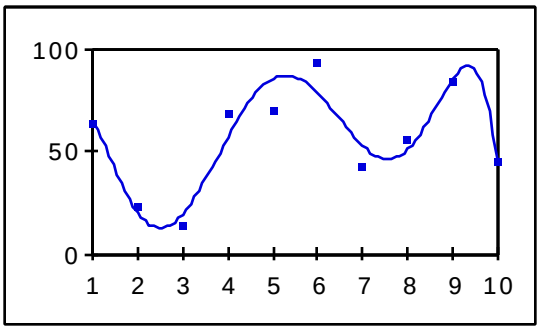



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 57 108  70 125 116 13 63 67 45 121 181
               189 201 166 115 105 91 75 99 72 33 29 69
                85 162 210 211 256 304 258 274 270 263;
               233 229 269 285 362 410 411 456 504 458
               474 470 463 257 308 270 325 316 213 263
               267 245 321 381 389 401 366 315 305 291
               275 299 272)
     ScatterChart()
     SymbolStyle(all;bullet;2)
     CurveFitting(1;linear)
     CurveFitting(2;7)
     CurveFittingOptions(1;1;;on;on)
     CurveFittingOptions(2;7;;on;on)
     MajorGridLineWidths(all;all;0) // Raster ausblenden
     GridFrame()
CloseDrawing()
```



Die Darstellung der berechneten Kurven erfolgt als Polygonzug. Standardmäßig werden dazu 25 Punkte verwendet. Mit dem Argument *punkteanzahl* kann die Anzahl der Polygonpunkte kontrolliert werden. So kann es bei Polynomen höherer Ordnung zweckmäßig sein, die Anzahl der Punkte zu erhöhen. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(64 23 14 69 70 94 43 56 85 45)
     ScatterChart()
     SymbolStyle(all;square;2;1;blue)
     CurveFitting(1;7)
     CurveFittingLineStyle(1;7;1;blue)
     CurveFittingOptions(1;7;;on;on;100)
     MajorGridLineWidths(all;all;0) // Raster ausblenden
     GridFrame()
CloseDrawing()
```



Fehlerbalken

Punkt-, Linien- und Balkendiagramme können durch Fehlerbalken ergänzt werden. Zur Berechnung und Darstellung der Fehlerbalken stehen vier Funktionen zur Verfügung.

ErrorBars(serienindex;achsenindex;fehlerrichtung;typ; wert1;wert2;wert3;wert4)

Durch die Funktion `ErrorBars()` können alle wichtigen Größen zur Berechnung der Fehlerbalken kontrolliert werden. Das erste Argument *serienindex* legt fest, auf welche Datenserie sich die Funktion bezieht. Mit dem zweiten Argument *achsenindex* kann bei zweidimensionalen Diagrammen die Richtung der Fehlerbalken vorgegeben werden. Wird kein Achsenindex vorgegeben oder *achsenindex=all* gesetzt, erfolgt die Berechnung der Fehlerbalken sowohl in x- als auch in y-Richtung. Mit dem 3. Argument *fehlerrichtung* kann festgelegt werden, ob nur die positiven Fehler (*fehlerrichtung=plus*) oder nur die negativen Fehler (*fehlerrichtung=minus*) oder beide Fehler (*fehlerrichtung=both*) dargestellt werden sollen. Standardmäßig werden beide Fehler dargestellt (*fehlerrichtung=both*). Die Art der Fehlerberechnung wird durch das Argument *typ* festgelegt. Die folgenden Fehlerberechnungen sind möglich:

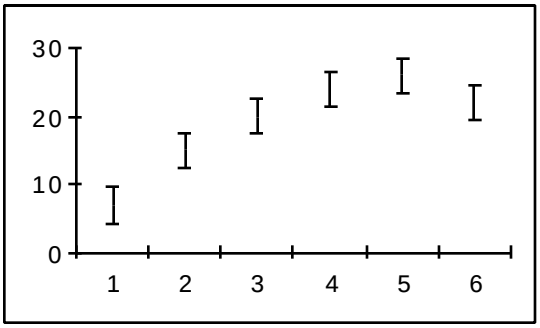
- Standardfehler: (*typ=stdError*)
- Standardabweichung: (*typ=stdDev*)
- Prozentuelle Fehler: (*typ=percent*)
- Konstante Fehler: (*typ=constant*)
- Benutzerdefinierte Fehlerlisten: (*typ=valueList*)

Wird *typ* nicht vorgegeben, so wird per default der Standardfehler berechnet. Benutzerdefinierte Fehlerlisten (*typ=valueList*) werden durch die Funktion `ErrorBarData()` übergeben. Die Bedeutung der Argumente *wert1* bis *wert4* ist abhängig von der Art der Fehlerberechnung:

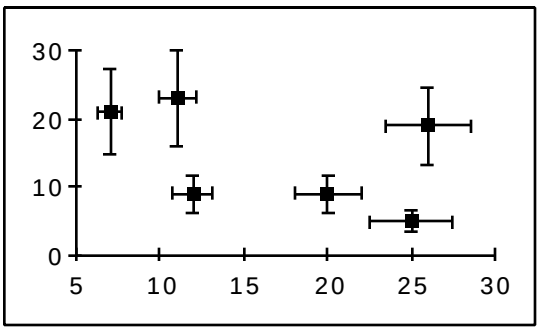
- Standardfehler:
  wert1...wert4 haben keine Bedeutung und werden ignoriert.
- Standardabweichung:
  wert1...wert4 sind die Standardabweichnungen, default: 1
- Prozentuelle Fehler:
  wert1...wert4 sind Prozentwerte, default: 5 [%]
- Konstante Fehler:
  wert1...wert4 sind konstante Werte, default: 1
- Benutzerdefinierte Fehlerlisten:
  wert1...wert4 haben keine Bedeutung und werden ignoriert.

Das Argument *wert1* bezieht sich auf den positiven Fehler in x-Richtung, das Argument *wert2* auf den negativen Fehler in x-Richtung, die Argumente *wert3* und *wert4* beziehen sich auf den positiven bzw. negativen Fehler in y-Richtung. Beispiele:

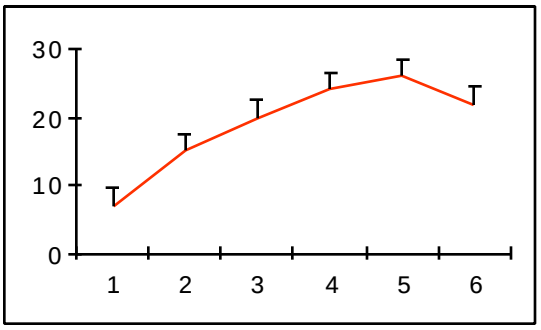
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 24 26 22)
     ScatterChart(;on)
     SymbolStyle(all;none)
     ErrorBars() // default: Standardfehler
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 12 20 26 25; // x-Werte
               21 23  9  9 19  5) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;square;4;1;black)
     ErrorBars(1;all;both;percent;10;10;30;30)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



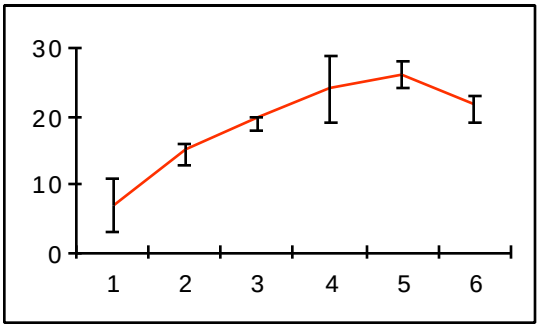
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 24 26 22)
     LineChart(;on)
     ErrorBars(1;;plus)     // default: Standardfehler
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



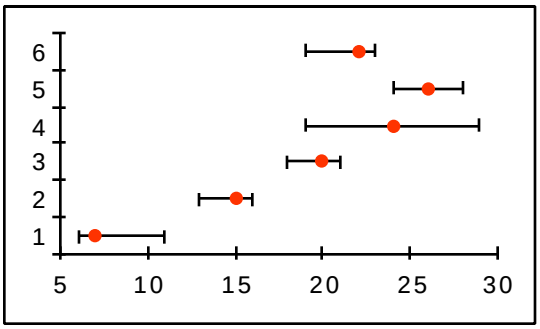
ErrorBarData(serienindex;werteliste1;werteliste2; werteliste3;werteliste4)

Mittels der Funktion `ErrorBarData()` können benutzerdefinierte Fehlerwerte übergeben werden. Korrespondierend dazu ist in der Funktion `ErrorBars()` das 4. Argument *typ=valueList* zu setzen, andernfalls wird die Funktion `ErrorBarData()` ignoriert. Das Argument *werteliste1* enthält die positiven Fehlerwerte in x-Richtung, das Argument *werteliste2* die negativen Fehlerwerte in x-Richtung. Analog repräsentieren die Argumente *werteliste3* und *werteliste4* die positiven bzw. negativen Fehlerwerte in y-Richtung. Die Werte einer Liste sind gleich wie in der Funktion `ChartData()` getrennt durch Leerzeichen einzugeben. Ist die Anzahl der übergebenen Fehlerwerte kleiner als die Anzahl der Datenpunkte, so werden die fehlenden Werte durch Nullen ergänzt. Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 24 26 22)
     LineChart(;on)
     ErrorBars(1;y;both;valueList)
     ErrorBarData(1;;;4 1 0 5 2 1;4 2 2 5 2 3)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 15 20 24 26 22)
     ScatterChart(horizontal;on)
     SymbolStyle(all;bullet;4;1;red)
     ErrorBars(1;;;valueList)
     ErrorBarData(1;4 1 1 5 2 1; 1 2 2 5 2 3)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```

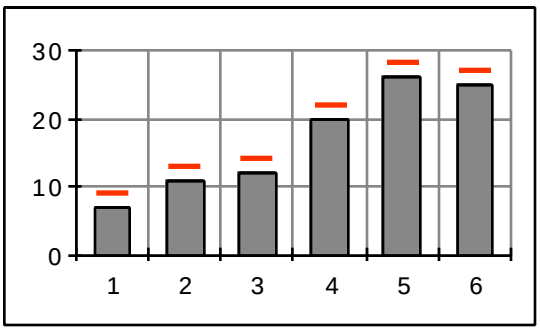


ErrorBarStyle(serienindex;achsenindex;nurMarkerZeigen; markerlänge;strichstärke;farbe;muster)

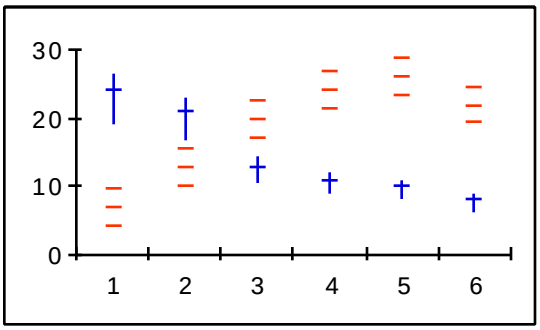
Mit der Funktion `ErrorBarStyle()` kann das Aussehen der Fehlerbalken kontrolliert werden. Das erste Argument *serienindex* legt fest, auf welche Datenserie sich die Funktion bezieht. Das zweite Argument *achsenindex* legt fest, auf welche Fehlerbalkenrichtung sich die Funktion bezieht. Durch Aktivieren des Arguments *nurMarkerZeigen=on* können die Verbindungslinien zwischen den Fehlermarkern unterdrückt werden. Die Länge der Fehlermarker (in Pixels) kann mit dem Argument *markerlänge* kontrolliert werden. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Fehlerbalken schwarz und 1 Pixel breit dargestellt.

Beispiele:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(7 11 12 20 26 25)
     BarChart()
     FillStyle(all;gray)
     ErrorBars(1;all;plus;constant;;;2)
     ErrorBarStyle(all;;on;13;2;red)
CloseDrawing()
```



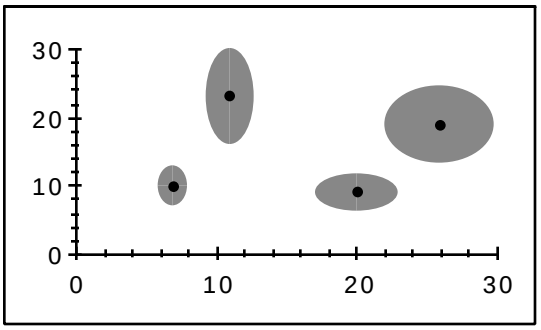

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 13 20 24 26 22; // 1.Serie
               24 21 13 11 10  8) // 2.Serie
     ScatterChart(;on)
     SymbolStyle(all;hBar;6)
     ErrorBars(1;y;both;stdError)
     ErrorBars(2;y;both;percent;;;10;20)
     ErrorBarStyle(1;y;on ;6;1;red)
     ErrorBarStyle(2;y;off;0;1;blue)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



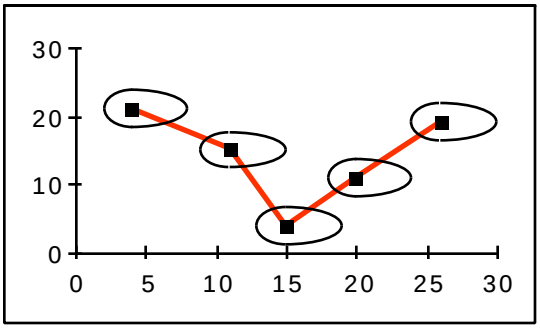
ErrorBarStyle2D(serienindex;form;füllfarbe;füllmuster; rahmenbreite;rahmenfarbe;rahmenmuster)

Durch die Funktion ErrorBarStyle2D() können bei zweidimensionalen Diagrammen die Fehlerwerte optional auch durch Ellipsen oder Rechtecke dargestellt werden. Das Argument *form* ermöglicht die Wahl zwischen elliptischen Fehlerflächen (*form=oval*) oder rechteckigen Fehlerflächen (*form=rect*). Wird *form* nicht vorgegeben, so werden standardmäßig Ellipsen zur Darstellung der Fehler verwendet. Die Fläche kann durch die Argumente *füllfarbe* und *füllmuster*, die Berandung durch die Argumente *rahmenbreite*, *rahmenfarbe* und *rahmenmuster* variiert werden. Standardmäßig werden Ellipsen durch einen 1 Pixel breiten, schwarzen Rand und ohne Füllung dargestellt. Beispiele:

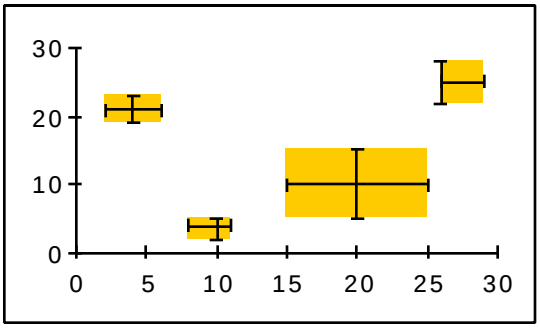
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 7 11 20 26; // x-Werte
               10 23  9 19) // y-Werte
     ScatterChart2D()
     Scaling(all;linear;0;30;3;5)
     SymbolStyle(all;bullet;3;1;black)
     ErrorBars(1;all;both;percent;15;15;30;30)
     ErrorBarStyle(1;all;on;0)
     ErrorBarStyle2D(1;oval;gray;black;0)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(4 11 15 20 26; 21 15 4 11 19)
     LineChart2D(symbol)
     LineStyle(1;poly;2)
     SymbolStyle(all;square;4;1;black)
     ErrorBars(1;x;both;constant;4;2)
     ErrorBars(1;y;both;stdError)
     ErrorBarStyle(1;all;on;0)
     ErrorBarStyle2D(1;oval)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```




```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData( 4 10 26 20; // x-Werte
               21  4 25 10) // y-Werte
     ScatterChart2D()
     SymbolStyle(all;none;3;1;black)
     ErrorBars(1;all;both;valueList)
     ErrorBarData(1;2 1 3 5; // positive x-Fehler
                    2 2 0 5; // negative x-Fehler
                    2 1 3 5; // positive y-Fehler
                    2 2 3 5) // negative y-Fehler
     ErrorBarStyle2D(1;rect;darkYellow;black;0)
     GridLocation(all;none) // kein Raster
CloseDrawing()
```


30
20
10
0
0
5
10
15
20
25
30

Grafische Basisfunktionen

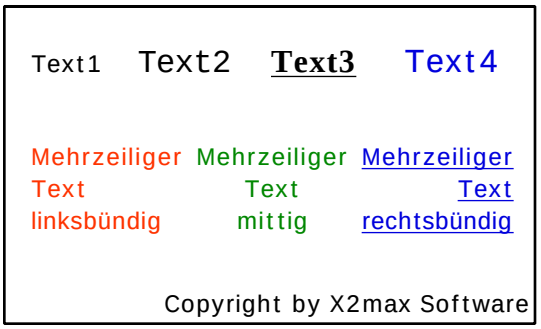
Zusätzlich zu den Diagrammfunktionen steht ein Satz von über 20 grafischen Basisfunktionen zum Zeichnen von Linien, Rechtecken, Ellipsen, Polygonen, Bitmaps, Texten etc. zur Verfügung. Diagramme können dadurch mit Texten, Logos usw. ergänzt werden. Es können aber auch allein aus den Basisfunktionen komplette Zeichnungen aufgebaut werden. Die Mächtigkeit und die Vielzahl der Basisfunktionen ermöglichen einen äußerst vielfältigen Einsatz.

Die Koordinaten beziehen sich bei allen Basisfunktionen auf die linke obere Ecke der Zeichnung. Falls Funktionen innerhalb eines Views angeführt werden, beziehen sich die Koordinaten auf die linke obere Ecke des Views. Views können auch dazu verwendet werden, mehrere Funktionen zu Gruppen zusammenzufassen. Die Gruppen können dann einfach durch Ändern der Viewpositionen auf der Zeichnung verschoben werden.

AddText(hPosition;vPosition;text;schrift;größe;stil; farbe;ausrichtung)

Bei linksbündiger Ausrichtung des Textes (default) wird durch *hPosition* der linke Rand des Textes festgelegt, bei mittiger Ausrichtung (*ausrichtung=center*) definiert *hPosition* die Mitte des Textes, bei rechtsbündiger Ausrichtung (*ausrichtung=right*) legt *hPosition* den rechten Rand des Textes fest. Mit dem Argument *vPosition* wird die vertikale Position der ersten Textzeile festgelegt, genauer gesagt, die Grundlinie der ersten Textzeile. Mehrzeilige Texte werden durch Einfügen eines Zeilenumbruchs "\n" erstellt. Beispiel:

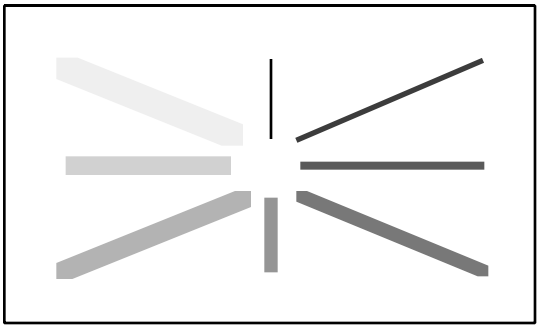
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddText( 10;25;"Text1")
     AddText( 50;25;"Text2";"Courier";12)
     AddText(100;25;"Text3";"Times";12;bold+underline)
     AddText(150;25;"Text4";;12;plain;blue)
     AddText(10;60;"Mehrzeiliger\nText\nlinksbündig";
             ;;;red)
     AddText(100;60;"Mehrzeiliger\nText\nmittig";
             ;;italic;green;center)
     AddText(190;60;"Mehrzeiliger\nText\nrechtsbündig";
             ;;underline;blue;right)
     AddText(60;115;"Copyright by X2max Software")
CloseDrawing()
```



AddLine(hStart;vStart;hEnde;vEnde;strichstärke;farbe; muster)

Eventuell benötigte Linien können durch die Funktion AddLine() hinzugefügt werden; wobei die Strichstärke, die Farbe und das Muster variiert werden können. Der Koordinatenursprung befindet sich in der linken, oberen Ecke der Zeichnung bzw. des Views. Beispiel:

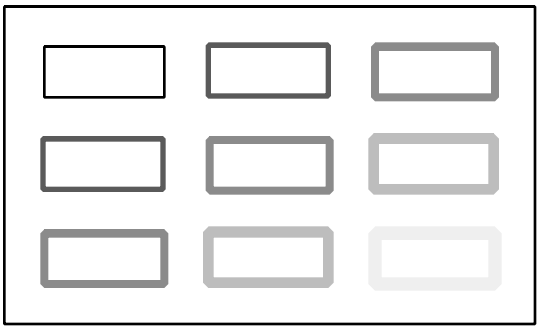
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddLine(100;50;100; 20;1;  0   0   0)
     AddLine(110;50;180; 20;2; 60  60  60)
     AddLine(110;60;180; 60;3; 90  90  90)
     AddLine(110;70;180;100;4;120 120 120)
     AddLine(100;70;100;100;5;150 150 150)
     AddLine( 90;70; 20;100;6;180 180 180)
     AddLine( 85;60; 20; 60;7;210 210 210)
     AddLine( 90;45; 20; 20;8;240 240 240)
CloseDrawing()
```



AddFrame(links;oben;breite;höhe;rahmenbreite;farbe; muster)

Die Funktion AddFrame() ermöglicht die Erstellung rechteckiger Rahmen, wobei die Rahmenbreite, die Farbe und das Muster variiert werden können. Beispiel:

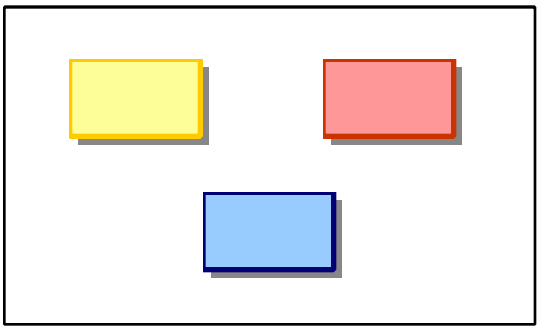
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     // linke Spalte
     AddFrame(15;15;46;20;1;  0   0   0)
     AddFrame(15;50;46;20;2; 90  90  90)
     AddFrame(15;85;46;20;3;140 140 140)

     // mittlere Spalte
     AddFrame(77;15;46;20;2; 90  90  90)
     AddFrame(77;50;46;20;3;140 140 140)
     AddFrame(77;85;46;20;4;190 190 190)

     // rechte Spalte
     AddFrame(139;15;46;20;3;140 140 140)
     AddFrame(139;50;46;20;4;190 190 190)
     AddFrame(139;85;46;20;5;240 240 240)
CloseDrawing()
```



AddRect(links;oben;breite;höhe;füllfarbe;füllmuster)

Die Funktion AddRecht() dient zur Erstellung rechteckiger Flächen, wobei die Farbe und das Muster variiert werden können. Beispiel:

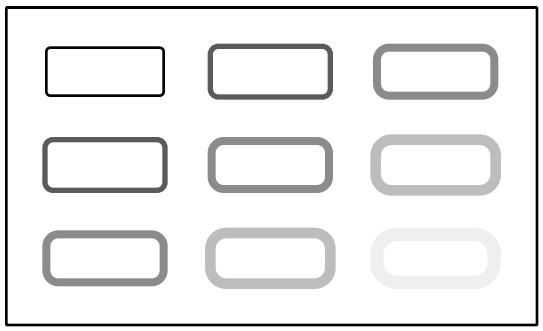
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     OpenView(25;20;53;33) // links oben
          AddRect(3;3;50;30;gray)
          AddRect(0;0;50;30;lightYellow)
          AddFrame(0;0;50;30;2;darkYellow)
     CloseView()

     OpenView(120;20;53;33) // rechts oben
          AddRect(3;3;50;30;gray)
          AddRect(0;0;50;30;lightRed)
          AddFrame(0;0;50;30;2;darkRed)
     CloseView()

     OpenView(75;70;53;33) // unten Mitte
          AddRect(3;3;50;30;gray)
          AddRect(0;0;50;30;lightBlue)
          AddFrame(0;0;50;30;2;darkBlue)
     CloseView()
CloseDrawing()
```



AddRoundFrame(links;oben;breite;höhe;hKrümmung; vKrümmung;rahmenbreite;farbe;muster)

Die Funktion AddRoundFrame() dient zur Erstellung von Rahmen mit abgerundeten Ecken, wobei die Rahmenbreite, die Farbe und das Muster variiert werden können. Die Stärke der Rundung wird durch die Argumente *hKrümmung* und *vKrümmung* kontrolliert. Je größer *hKrümmumg* bzw. *vKrümmung* desto stärker die Rundung. Bei *hKrümmung=0* bzw. *vKrümmung=0* verschwindet die Rundung und AddRoundFrame() geht in AddFrame() über. Beispiel:

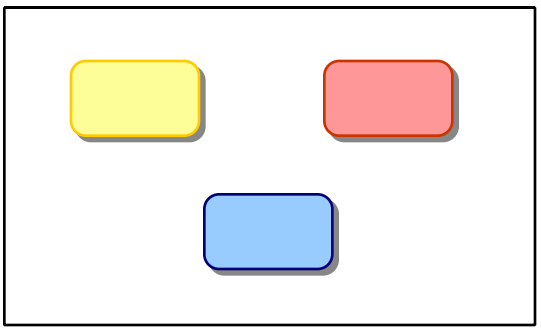
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     // linke Spalte
     AddRoundFrame(15;15;46;20; 4; 4;1;  0   0   0)
     AddRoundFrame(15;50;46;20; 8; 8;2; 90  90  90)
     AddRoundFrame(15;85;46;20;12;12;3;140 140 140)

     // mittlere Spalte
     AddRoundFrame(77;15;46;20; 8; 8;2; 90  90  90)
     AddRoundFrame(77;50;46;20;12;12;3;140 140 140)
     AddRoundFrame(77;85;46;20;16;16;4;190 190 190)

     // rechte Spalte
     AddRoundFrame(139;15;46;20;12;12;3;140 140 140)
     AddRoundFrame(139;50;46;20;16;16;4;190 190 190)
     AddRoundFrame(139;85;46;20;20;20;5;240 240 240)
CloseDrawing()
```



AddRoundRect(links;oben;breite;höhe;hKrümmung; vKrümmung;füllfarbe;füllmuster)

Die Funktion AddRoundRect() ermöglicht die Erstellung von abgerundeten Rechteckflächen, wobei die Farbe und das Muster variiert werden können. Die Stärke der Rundung wird durch die Argumente *hKrümmung* und *vKrümmung* kontrolliert. Je größer *hKrümmumg* bzw. *vKrümmung* desto stärker die Rundung. Bei *hKrümmung=0* bzw. *vKrümmung=0* verschwindet die Rundung und AddRoundRect() geht in AddRect() über. Beispiel:

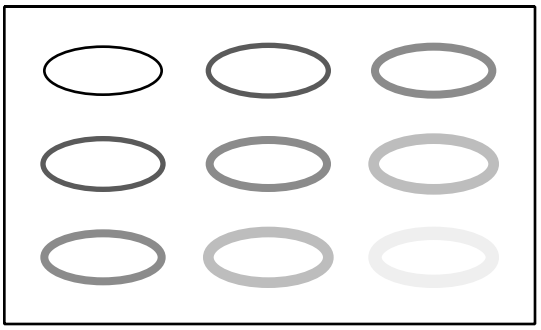
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     OpenView(25;20;52;32) // links oben
          AddRoundRect(2;2;50;30;;;gray)
          AddRoundRect(0;0;50;30;;;lightYellow)
          AddRoundFrame(0;0;50;30;;;1;darkYellow)
     CloseView()

     OpenView(120;20;52;32) // rechts oben
          AddRoundRect(2;2;50;30;;;gray)
          AddRoundRect(0;0;50;30;;;lightRed)
          AddRoundFrame(0;0;50;30;;;1;darkRed)
     CloseView()

     OpenView(75;70;52;32) // unten Mitte
          AddRoundRect(2;2;50;30;;;gray)
          AddRoundRect(0;0;50;30;;;lightBlue)
          AddRoundFrame(0;0;50;30;;;1;darkBlue)
     CloseView()
CloseDrawing()
```



AddEllipse(links;oben;breite;höhe;strichstärke;farbe; muster)

Die Funktion AddEllipse() ermöglicht die Erstellung ovaler Rahmen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Beispiel:

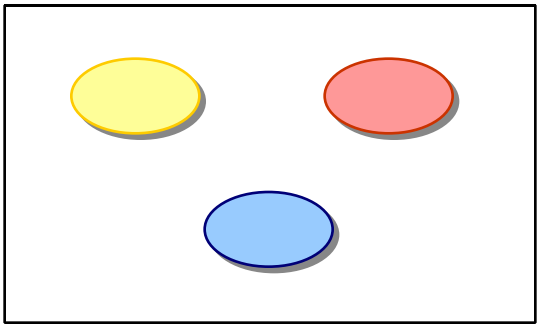
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     // linke Spalte
     AddEllipse(15;15;46;20;1;  0   0   0)
     AddEllipse(15;50;46;20;2; 90  90  90)
     AddEllipse(15;85;46;20;3;140 140 140)

     // mittlere Spalte
     AddEllipse(77;15;46;20;2; 90  90  90)
     AddEllipse(77;50;46;20;3;140 140 140)
     AddEllipse(77;85;46;20;4;190 190 190)

     // rechte Spalte
     AddEllipse(139;15;46;20;3;140 140 140)
     AddEllipse(139;50;46;20;4;190 190 190)
     AddEllipse(139;85;46;20;5;240 240 240)
CloseDrawing()
```



AddOval(links;oben;breite;höhe;füllfarbe;füllmuster)

Die Funktion AddOval() ermöglicht die Erstellung ovaler Flächen, wobei die Farbe und das Muster variiert werden können. Beispiel:

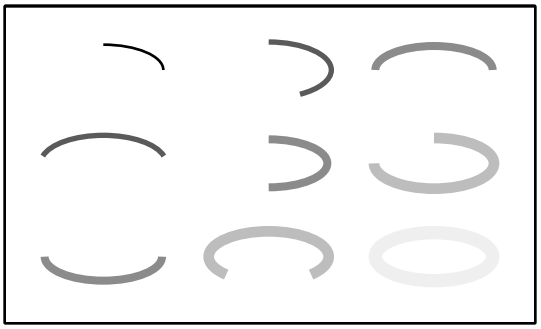
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     OpenView(25;20;52;32) // links oben
          AddOval(2;2;50;30;gray)
          AddOval(0;0;50;30;lightYellow)
          AddEllipse(0;0;50;30;1;darkYellow)
     CloseView()

     OpenView(120;20;52;32) // rechts oben
          AddOval(2;2;50;30;gray)
          AddOval(0;0;50;30;lightRed)
          AddEllipse(0;0;50;30;1;darkRed)
     CloseView()

     OpenView(75;70;53;33) // unten Mitte
          AddOval(2;2;50;30;gray)
          AddOval(0;0;50;30;lightBlue)
          AddEllipse(0;0;50;30;1;darkBlue)
     CloseView()
CloseDrawing()
```



**AddArc(links;oben;breite;höhe;startwinkel;
öffnungswinkel;strichstärke;farbe;muster)**

Die Funktion AddArc() ermöglicht die Erstellung ovaler Bögen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Start- und Öffnungswinkel sind im Bereich von ± 360 Grad anzugeben; dabei ist *startwinkel=0* oben (12 Uhr), *startwinkel=90* rechts (3 Uhr) und *startwinkel=-90* links (9 Uhr). Ist der Öffungswinkel positiv, wird der Bogen im Uhrzeigersinn aufgetragen, ist der Öffnungswinkel negativ, wird der Bogen gegen den Uhrzeigersinn aufgetragen. Beispiel:

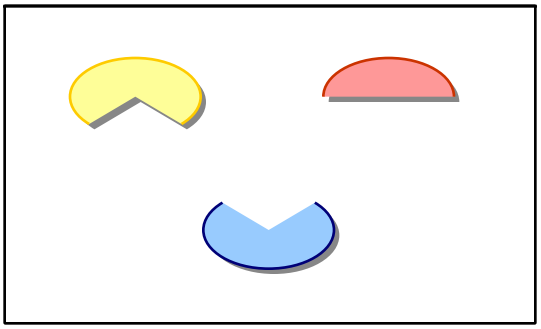
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     // linke Spalte
     AddArc(15;15;46;20;  0; 90;1;  0   0   0)
     AddArc(15;50;46;20;-75;150;2; 90  90  90)
     AddArc(15;85;46;20; 90;180;3;140 140 140)

     // mittlere Spalte
     AddArc(77;15;46;20;   0;150;2; 90  90  90)
     AddArc(77;50;46;20;   0;180;3;140 140 140)
     AddArc(77;85;46;20;-135;270;4;190 190 190)

     // rechte Spalte
     AddArc(139;15;46;20;-90;180;3;140 140 140)
     AddArc(139;50;46;20;  0;270;4;190 190 190)
     AddArc(139;85;46;20;  0;360;5;240 240 240)
CloseDrawing()
```



AddSlice(links;oben;breite;höhe;startwinkel; öffnungswinkel;füllfarbe;füllmuster)

Die Funktion AddSlice() dient zur Erstellung oval begrenzter Sektoren, wobei die Farbe und das Muster variiert werden können. Die Erklärung der Argumente *startwinkel* und *öffnungswinkel* findet sich bei der Funktion AddArc(). Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     OpenView(25;20;52;32) // links oben
          AddSlice(2;2;50;30;-135;270;gray)
          AddSlice(0;0;50;30;-135;270;lightYellow)
          AddArc(0;0;50;30;-135;270;1;darkYellow)
     CloseView()

     OpenView(120;20;52;32) // rechts oben
          AddSlice(2;2;50;30;-90;180;gray)
          AddSlice(0;0;50;30;-90;180;lightRed)
          AddArc(0;0;50;30;-90;180;1;darkRed)
     CloseView()

     OpenView(75;70;53;33) // unten Mitte
          AddSlice(2;2;50;30;45;270;gray)
          AddSlice(0;0;50;30;45;270;lightBlue)
          AddArc(0;0;50;30;45;270;1;darkBlue)
     CloseView()
CloseDrawing()
```



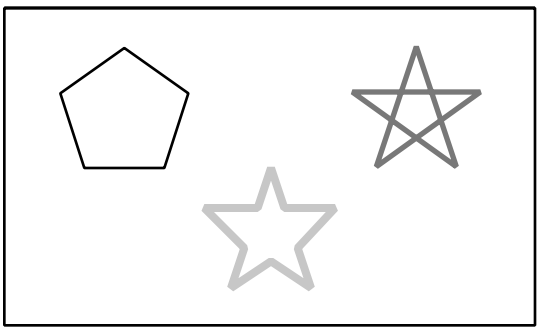
AddPolyline(koordReihenfolge;koordListe;strichstärke; farbe;muster)

Die Funktion AddPolyline() ermöglicht die Erstellung von Linienzügen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Das Argument *koordReihenfolge* ermöglicht die Eingabe der Koordinaten auf zwei Arten: Entweder werden zuerst alle x-Koordinaten und dann alle y-Koordinaten angeführt (*koordReihenfolge=xxyy*) oder es werden die x- und y-Koordinaten des 1. Polygonpunktes angeführt, dann die x- und y-Koordinaten des 2. Polygonpunktes, usw. (*koordReihenfolge=xyxy*). Alle Koordinatenwerte sind getrennt durch Leerzeichen, Tabs oder Zeilenumbrüche einzugeben. Die folgenden zwei Funktionen definieren den gleichen Linienzug:

```
(1)  AddPolyline(xxyy;45 69 60 30 21 45
                       15 32 60 60 32 15)
(2)  AddPolyline(xyxy;45 15
                       69 32
                       60 60
                       30 60
                       21 32
                       45 15)
```

Beispiel:

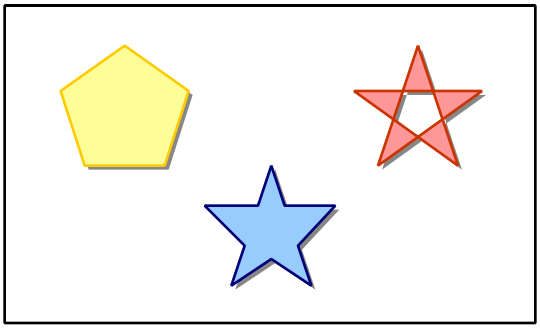
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddPolyline(xyxy;45 15  69 32  60 60  30 60
                      21 32  45 15; 1)
     AddPolyline(xyxy;155 15  140 60  179 32  131 32
                      170 60  155 15; 2;120 120 120)
     AddPolyline(xyxy;100  60   95 75   75  75   90 90
                       85 105  100 95  115 105  110 90
                      124  75  105 75  100  60;
                      3;200 200 200)
CloseDrawing()
```



AddPolygon(koordReihenfolge;koordListe;füllfarbe; füllmuster)

Die Funktion AddPolygon() ermöglicht die Darstellung polygonal begrenzter Flächen, wobei die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

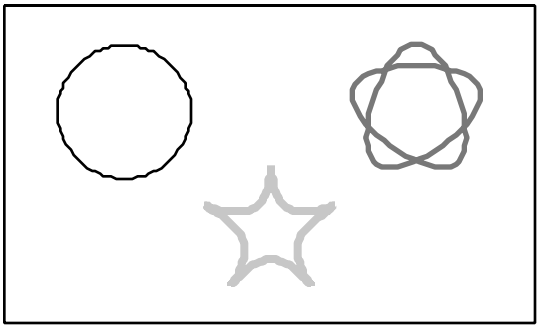
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddPolygon(xyxy;47 17  71 34  62 62  32 62
                     23 34  47 17; gray)
     AddPolygon(xyxy;45 15  69 32  60 60  30 60
                     21 32  45 15; lightYellow)
     AddPolyline(xyxy;45 15  69 32  60 60  30 60
                      21 32  45 15; 1; darkYellow)
     AddPolygon(xyxy;157 17  142 62  181 34  133 34
                     172 62  157 17; gray)
     AddPolygon(xyxy;155 15  140 60  179 32  131 32
                     170 60  155 15; lightRed)
     AddPolyline(xyxy;155 15  140 60  179 32  131 32
                      170 60  155 15; 1; darkRed)
     AddPolygon(xyxy;102  62   97 77   77  77   92 92
                      87 107  102 97  117 107  112 92
                     126  77  107 77  102  62;
                     gray)
     AddPolygon(xyxy;100  60   95 75   75  75   90 90
                      85 105  100 95  115 105  110 90
                     124  75  105 75  100  60;
                     lightBlue)
     AddPolyline(xyxy;100  60   95 75   75  75   90 90
                       85 105  100 95  115 105  110 90
                      124  75  105 75  100  60;
                      1; darkBlue)
CloseDrawing()
```



AddSmoothPolyline(koordReihenfolge;koordListe; strichstärke;farbe;muster)

Die Funktion AddSmoothPolyline() ermöglicht die Erstellung von geglätteten Linienzügen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

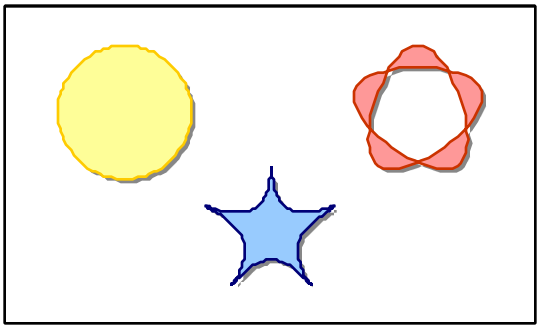
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddSmoothPolyline(xyxy;45 15  69 32  60 60  30 60
                            21 32  45 15;1)
     AddSmoothPolyline(xyxy;155 15  140 60  179 32  131 32
                            170 60  155 15; 2;120 120 120)
     AddSmoothPolyline(xyxy;100  60  95 75  75  75  90 90
                             85 105 100 95 115 105 110 90
                            124  75 105 75 100  60;
                            3;200 200 200)
CloseDrawing()
```



AddSmoothPolygon(koordReihenfolge;koordListe;füllfarbe; füllmuster)

Die Funktion AddSmoothPolygon() ermöglicht die Darstellung von glatt berandeten Flächen, wobei die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddSmoothPolygon(xyxy;47 17  71 34  62 62  32 62
                           23 34  47 17; gray)
     AddSmoothPolygon(xyxy;45 15  69 32  60 60  30 60
                           21 32  45 15; lightYellow)
     AddSmoothPolyline(xyxy;45 15  69 32  60 60  30 60
                            21 32  45 15; 1;darkYellow)
     AddSmoothPolygon(xyxy;157 17  142 62  181 34  133 34
                           172 62  157 17; gray)
     AddSmoothPolygon(xyxy;155 15  140 60  179 32  131 32
                           170 60  155 15; lightRed)
     AddSmoothPolyline(xyxy;155 15  140 60  179 32  131 32
                            170 60  155 15; 1;darkRed)
     AddSmoothPolygon(xyxy;102  62   97 77   77  77  92 92
                            87 107  102 97  117 107 112 92
                           126  77  107 77  102  62; gray)
     AddSmoothPolygon(xyxy;100  60   95 75   75  75  90 90
                            85 105  100 95  115 105 110 90
                           124  75  105 75  100  60;
                           lightBlue)
     AddSmoothPolyline(xyxy;100  60   95 75  75  75  90 90
                             85 105  100 95 115 105 110 90
                            124  75  105 75 100  60;
                            1;darkBlue)
CloseDrawing()
```

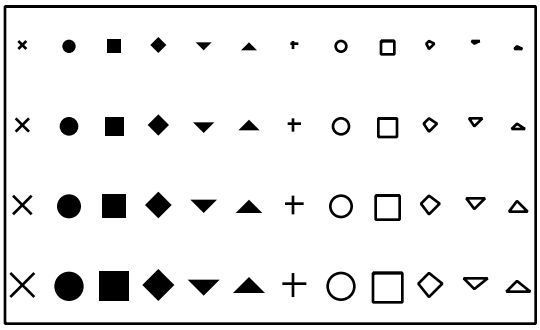


AddSymbol(hPosition;vPosition;symboltyp;symbolgröße; strichstärke;farbe;muster)

Die Funktion AddSymbol() ermöglicht die Darstellung von Symbolen, wobei die Symbolgröße, die Strichstärke, die Farbe und das Muster variiert werden können. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     // 1.Zeile
     AddSymbol(  7;15; 1;4)
     AddSymbol( 24;15; 2;4)
     AddSymbol( 41;15; 3;4)
     AddSymbol( 58;15; 4;4)
     AddSymbol( 75;15; 5;4)
     AddSymbol( 92;15; 6;4)
     AddSymbol(109;15; 7;4)
     AddSymbol(126;15; 8;4)
     AddSymbol(143;15; 9;4)
     AddSymbol(160;15;10;4)
     AddSymbol(177;15;11;4)
     AddSymbol(193;15;12;4)

     // 2.Zeile
     AddSymbol(  7;45; 1;6)
     AddSymbol( 24;45; 2;6)
     AddSymbol( 41;45; 3;6)
     AddSymbol( 58;45; 4;6)
     AddSymbol( 75;45; 5;6)
     AddSymbol( 92;45; 6;6)
     AddSymbol(109;45; 7;6)
     AddSymbol(126;45; 8;6)
     AddSymbol(143;45; 9;6)
     AddSymbol(160;45;10;6)
     AddSymbol(177;45;11;6)
     AddSymbol(193;45;12;6)
```

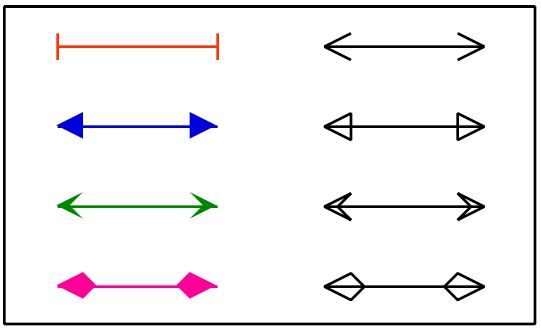
```
        // 3.Zeile
        AddSymbol(  7;75; 1;8)
        AddSymbol( 24;75; 2;8)
        AddSymbol( 41;75; 3;8)
        AddSymbol( 58;75; 4;8)
        AddSymbol( 75;75; 5;8)
        AddSymbol( 92;75; 6;8)
        AddSymbol(109;75; 7;8)
        AddSymbol(126;75; 8;8)
        AddSymbol(143;75; 9;8)
        AddSymbol(160;75;10;8)
        AddSymbol(177;75;11;8)
        AddSymbol(193;75;12;8)

        // 4.Zeile
        AddSymbol(  7;105; 1;10)
        AddSymbol( 24;105; 2;10)
        AddSymbol( 41;105; 3;10)
        AddSymbol( 58;105; 4;10)
        AddSymbol( 75;105; 5;10)
        AddSymbol( 92;105; 6;10)
        AddSymbol(109;105; 7;10)
        AddSymbol(126;105; 8;10)
        AddSymbol(143;105; 9;10)
        AddSymbol(160;105;10;10)
        AddSymbol(177;105;11;10)
        AddSymbol(193;105;12;10)

CloseDrawing()
```



**AddArrow(hStart;vStart;hEnde;vEnde;strichstärke;farbe;
muster;pfeilposition;pfeillänge;pfeilbreite;
pfeilkerbe;istPfeilHohl)**

Die Funktion `AddArrow()` ermöglicht die Darstellung von Pfeilen. Dabei können sowohl die Strichstärke, die Farbe, das Muster und die Pfeilspitze variiert werden. Weiters kann durch das Argument *pfeilposition* die Pfeilspitze am Ende (default), am Anfang oder auch beidseitig angeordnet werden. Beispiel:

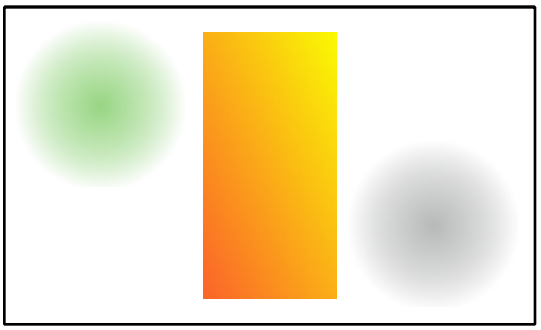
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)

     AddArrow( 20; 15; 80; 15;1;red;;begin+end;0;10;0)
     AddArrow(120; 15;180; 15;1;;;begin+end;10;10;10)
     AddArrow( 20; 45; 80; 45;1;blue;;begin+end;10;10)
     AddArrow(120; 45;180; 45;1;;;begin+end;10;10;0;on)
     AddArrow( 20; 75; 80; 75;1;green;;begin+end;10;10;5)
     AddArrow(120; 75;180; 75;1;;;begin+end;10;10;5;on)
     AddArrow( 20;105;80;105;1;purple;;begin+end;10;10;-5)
     AddArrow(120;105;180;105;1;;;begin+end;10;10;-5;on)

CloseDrawing()
```



AddPicture(links;oben;breite;höhe;quelle;name)

Mittels der Funktion `AddPicture()` ist es möglich, eine Zeichnung durch ein Bild wie z.B. einem Firmenlogo zu ergänzen. Dabei werden drei verschiedene Bildquellen unterstützt: die Zwischenablage (*quelle=clipboard*), eine Datei (*quelle=file*) oder eine Resource (*quelle=resource*). Die Argumente *quelle* und *name* werden ausführlich im Zuge der Funktion `PictureStyle()` im Abschnitt *Stile* erläutert. Die linke obere Ecke des Bildes wird entsprechend der Argumente *links* und *oben* platziert. Der Koordinatenursprung befindet sich in der linken, oberen Ecke der Zeichnung bzw. des Views. Die Argumente *breite* und *höhe* dienen zum Skalieren des Bildes. Wird die Breite bzw. die Höhe nicht vorgegeben, so wird zur Darstellung die tatsächliche Breite bzw. Höhe des Bildes verwendet. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddPicture(75;10;50;100;resource;"32")
     AddPicture(  5; 5;;;resource;"37")
     AddPicture(130;50;;;resource;"33")
CloseDrawing()
```

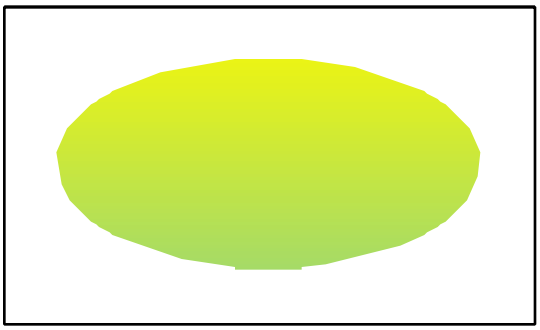


Die Sichtbarkeit der Basisfunktionen kann durch Anführen sogenannter Clippingfunktionen auf eine bestimmte Fläche (Clippingfläche) eingeschränkt werden.
Zur Zeit werden die Clippingfunktionen nur unter MacOS/X unterstützt. Die folgenden Funktionen können zur Festlegung der Clippingfläche herangezogen werden:

AddClipRect(typ;links;oben;breite;höhe)
AddClipRoundRect(typ;links;oben;breite;höhe;hKrümmung;
vKrümmung)
AddClipOval(typ;links;oben;breite;höhe)
AddClipSlice(typ;links;oben;breite;höhe;startwinkel;
öffnungswinkel)
AddClipPolygon(typ;koordReihenfolge;koordListe)
AddClipSmoothPolygon(typ;koordReihenfolge;koordListe)

Der Aufbau der Clippingfunktionen und die Bedeutung der Argumente sind identisch mit den Basisfunktionen. Das Argument *typ* ermöglicht die Kombination mehrerer Clippingfunktionen. Standardmäßig wird jede Clippingfunktion mit der aktuellen Clippingfläche geschnitten (*typ=sect*), d.h. die neue Clippingfläche kann nie größer werden als die zuvor vorhandene Clippingfläche. Beispiel:

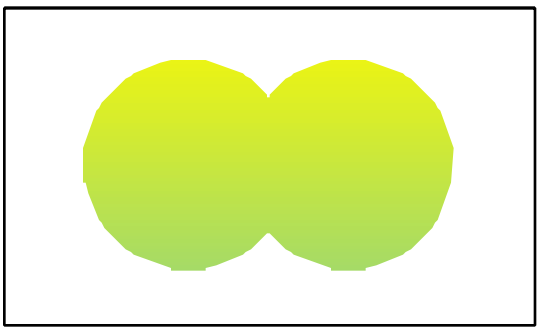
```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddClipOval(sect;20;20;160;80)
     AddPicture(0;0;200;120;resource;"7")
CloseDrawing()
```



Clippingflächen werden addiert, falls *typ=union* gesetzt wird; das heißt die vorhandene Clippingfläche wird um die durch die Clippingfunktion definierte Fläche erweitert.

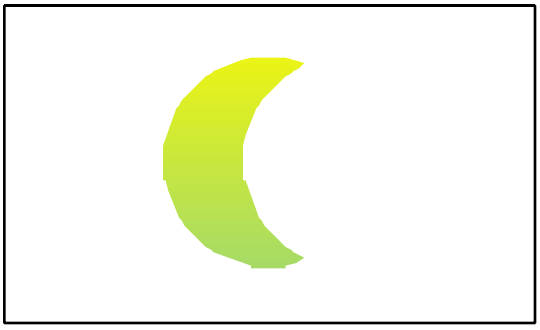
Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddClipOval(sect;30;20;80;80)
     AddClipOval(union;90;20;80;80)
     AddPicture(0;0;200;120;resource;"7")
CloseDrawing()
```



Falls *typ=diff* gesetzt wird, ergibt sich die neue Clippingfläche aus der Differenz zwischen der vorhandenen Clippingfläche und der durch die Clippingfunktion definierten Fläche. Beispiel:

```
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     AddClipOval(sect;60;20;80;80)
     AddClipOval(diff;90;20;80;80)
     AddPicture(0;0;200;120;resource;"7")
CloseDrawing()
```



AddClipReset()

Die Funktion `AddClipReset()` setzt die Clippingfläche auf die Breite und Höhe der Zeichnung zurück oder, falls die Funktion `AddClipReset()` innerhalb eines Views angeführt wird, auf die Breite und Höhe des Views.

Ausgabe

Ausgabefunktionen ermöglichen das Sichern einer Zeichnung in eine Datei. Dabei stehen zur Zeit die folgenden Formate zur Verfügung:

- MacOS/X: PICT, JPEG, PNG, BMP, TIFF
- Windows: EMF, GIF, JPEG, PNG, BMP, TIFF

PICT und EMF-Format sind plattformspezifische Vektorformate mit hoher Druckqualität, im Gegensatz zu GIF, JPG, PNG, BMP und TIFF-Format. Letztere stellen reine Bitmap-Formate dar, mit geringer Auflösung (z.B. 72dpi), können dafür aber plattformübergreifend verwendet werden.

Wichtig:
Um unter MacOS/X Dateien im JPEG-, PNG-, BMP- oder TIFF-Format erstellen zu können, muß QuickTime installiert und aktiviert sein. Die neueste QuickTime-Version kann unter dem folgenden Link heruntergeladen werden:
<http://www.apple.com/quicktime/download>

Wird keine Ausgabefunktion angeführt, so wird standardmäßig die Zeichnung in die Zwischenablage kopiert. Beispiel:

```
// Die Zeichnung wird in die Zwischenablage kopiert
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
```

Alle Ausgabefunktionen können an beliebiger Stelle, das heißt sowohl innerhalb als auch außerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()` angeführt werden.

SendToClipboard()

Die Funktion `SendToClipboard()` besitzt keine Argumente und muß nur dann angeführt werden, wenn eine Zeichnung sowohl als Datei gesichert als auch in die Zwischenablage kopiert werden soll. Wird eine Zeichnung nicht als Datei gesichert, dann wird diese automatisch in die Zwischenablage kopiert, so dass die Funktion `SendToClipboard()` nicht notwendig ist.

Beispiel 1:

```
// Die Zeichnung wird automatisch
// in die Zwischenablage kopiert.
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
SendToClipboard() // nicht notwendig
```

Beispiel 2:

```
// Die Zeichnung wird sowohl als PNG Datei gesichert
// als auch in die Zwischenablage kopiert.
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
SaveAsPNGFile("Zeichnung.png")
SendToClipboard()
```

Beispiel 3:

```
// Die Zeichnung wird NICHT in die Zwischenablage kopiert
OpenDrawing(200;120)
     AddFrame(0;0;200;120)
     ChartData(3 2 1 5 3)
     BarChart()
CloseDrawing()
SaveAsPNGFile("Zeichnung.png")
```

SaveAsPICTFile(dateiname;dateiflag;dateityp)

Die Funktion `SaveAsPICTFile()` steht nur unter MacOS/X zur Verfügung und ermöglicht das Sichern einer Zeichnung in eine PICT Datei. Unter Windows wird die Funktion `SaveAsPICTFile()` ignoriert. Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad definiert. Unter MacOS/X ist das Trennzeichen im Dateipfad ein Doppelpunkt ":", *ohne* zusätzliche Leerzeichen vor und nach dem Doppelpunkt. Zum Beispiel:

```
SaveAsPictFile("Macintosh HD:Bilder:Zeichnung")
```

Wird kein Pfad, sondern nur der Dateiname vorgegeben, so wird die Datei in denjenigen Ordner abgelegt, in dem sich die FileMaker Pro Applikation befindet. Ist eine Datei mit dem gleichen Namen bereits

vorhanden, so stehen mit dem zweiten Argument *dateiflag* folgende Optionen zur Verfügung:

- *dateiflag=addCounter*: (default)
  Standardmäßig wird ein Zähler an den Dateinamen angefügt, z.B. Diagramm-1, Diagramm-2, usw.
- *dateiflag=replace*:
  Die bereits vorhandene Datei wird überschrieben.
- *dateiflag=throwError*:
  Abbruch mit der Fehlermeldung *"Datei existiert bereits"*.

Mit dem dritten Argument *dateityp* kann optional unter MacOS/X noch der Dateityp (file creator) definiert werden. Damit wird festgelegt, mit welcher Applikation die PICT-Datei standardmäßig, d.h. beim Doppelklicken auf die Datei, geöffnet werden soll. Wenn nicht anders vorgegeben, wird die erstellte PICT-Datei als PictureViewer-Dokument (Dateityp `"ogle"`) weggeschrieben. Unter Windows wird das Argument *dateityp* ignoriert. Beispiele:

```
SaveAsPICTFile("Diagramm.pct")
SaveAsPICTFile("Chart";replace)
SaveAsPICTFile("Macintosh HD:FileMaker Pro:Diagramm")
SaveAsPICTFile("Plot.pct";;"8BIM") // Sichern als
                                   //"Photoshop"-Datei.
```

SaveAsEMFFile(dateiname;dateiflag)

Die Funktion `SaveAsEMFFile()` steht nur unter Windows zur Verfügung und ermöglicht das Sichern einer Zeichnung in eine EMF-Datei (Enhanced Metafile Format). Unter MacOS/X wird die Funktion `SaveAsEMFFile()` ignoriert. Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad definiert. Unter Windows ist das Trennzeichen im Dateipfad ein Backslash "\", ohne zusätzliche Leerzeichen davor und danach. Zu beachten ist, dass Sonderzeichen wie das Dateipfad-Trennzeichen "\" durch einen vorangestellten Backslash "\\" eingegeben werden müssen. (Siehe *xmReferenz.pdf*, Abschnitt *Diverses*). Zum Beispiel:

```
SaveAsEMFFile("C:\\Programme\\Plots\\Plot-1.emf")
```

Ist eine Datei mit dem gleichen Namen bereits vorhanden, so stehen mit dem zweiten Argument *dateiflag* folgende Optionen zur Verfügung:

- *dateiflag=addCounter*: (default)
  Standardmäßig wird ein Zähler an den Dateinamen angefügt, z.B. Diagramm-1.emf, Diagramm-2.emf, usw.
- *dateiflag=replace*:
  Die bereits vorhandene Datei wird überschrieben.
- *dateiflag=throwError*:
  Abbruch mit der Fehlermeldung *"Datei existiert bereits"*.

Beispiele:

```
SaveAsEMFFile("Diagramm.emf")
SaveAsEMFFile("Chart.emf";replace)
SaveAsEMFFile("C:\\Programme\\FileMaker\\Grafik.emf")
```

SaveAsGIFFile(dateiname;dateiflag)

Die Funktion SaveAsGIFFile() steht zur Zeit nur unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsEMFFile(). Zum Beispiel:

```
SaveAsGIFFile("Diagramm.gif")
SaveAsGIFFile("Chart.gif";replace)
SaveAsGIFFile("C:\\Programme\\FileMaker\\Grafik.gif")
```

SaveAsJPGFile(dateiname;dateiflag;dateityp;qualität)

Die Funktion SaveAsJPGFile() steht sowohl unter MacOS/X als auch unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPICTFile(). Zusätzlich kann mit dem 4. Argument *qualität* die Bildqualität der JPEG-Datei kontrolliert werden. Dazu stehen die folgenden fünf Konstanten zur Verfügung:

Konstante	*Wert*	*Anmerkung*
min	1	
low	2	
normal	3	(default)
high	4	
max	5	

Hohe Bildqualität bedeutet größere Dateien zufolge geringerer Komprimierung und umgekehrt, niedrige Bildqualität führt zu kleineren Dateien bei starker Komprimierung der Bildinformation. Wenn nicht anders vorgegeben, wird die erstellte JPEG-Datei mit *qualität=normal* weggeschrieben. Beispiele:

```
SaveAsJPGFile("Chart1.jpg")
SaveAsJPGFile("Diagramm.jpg";replace;;high)
SaveAsJPGFile("Macintosh HD:FileMaker Pro:Chart.jpg")
SaveAsJPGFile("C:\\Programme\\Plots\\Grafik.jpg";;;max)
```

SaveAsPNGFile(dateiname;dateiflag;dateityp)

Die Funktion SaveAsPNGFile() steht sowohl unter MacOS/X als auch unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPICTFile(). Zum Beispiel:

```
SaveAsPNGFile("Diagramm.png")
SaveAsPNGFile("Chart.png";replace)
SaveAsPNGFile("Macintosh HD:Programme:Plots:Grafik.png)
SaveAsPNGFile("C:\\Programme\\Plots\\Grafik.png")
```

SaveAsBMPFile(dateiname;dateiflag;dateityp)

Die Funktion SaveAsBMPFile() steht sowohl unter Windows als auch unter MacOS/X zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPICTFile().

Beispiele:

```
SaveAsBMPFile("Diagramm.bmp")
SaveAsBMPFile("Chart.bmp";replace)
SaveAsBMPFile("Macintosh HD:Programme:Plots:Grafik.bmp)
SaveAsBMPFile("C:\\Programme\\Plots\\Grafik.bmp")
```

SaveAsTIFFile(dateiname;dateiflag;dateityp)

Die Funktion SaveAsTIFFile() steht sowohl unter Windows als auch unter MacOS/X zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPICTFile().

Beispiele:

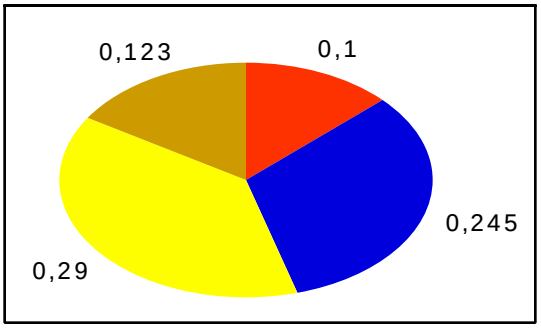
```
SaveAsTIFFile("Diagramm.tiff")
SaveAsTIFFile("Chart.tiff";replace)
SaveAsTIFFile("Macintosh HD:Programme:Plots:Grafik.tif")
SaveAsTIFFile("C:\\Programme\\Plots\\Grafik.tif")
```

Diverses

SetDecimalPoint(zeichen)

Durch die Funktion `SetDecimalPoint()` kann das Dezimalpunkt-Zeichen kontrolliert werden. `SetDecimalPoint()` ist eine "globale" Funktion, d.h. alle in einer Zeichung dargestellten Dezimalzahlen sind davon betroffen, unabhängig ob es sich dabei um Achsenbeschriftungen, Beschriftungen von Diagrammwerten, etc. handelt, auch unabhängig von eventuell vorhandenen Formatanweisungen. Beispiel:

```
OpenDrawing(200;120)
     SetDecimalPoint(",")
     AddFrame(0;0;200;120)
     ChartData(0.1 0.245 0,29 0,123)
     PieChart(label+oval)
     BorderStyle(all;none)
CloseDrawing()
```

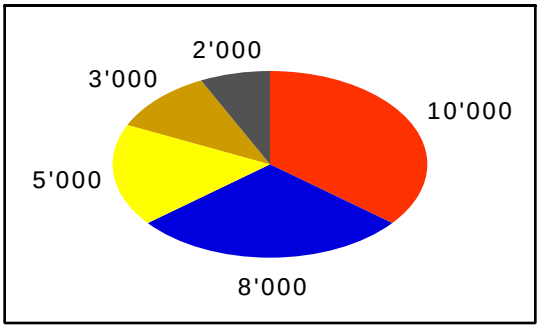


Die Eingabe von Dezimalzahlen wird durch die Funktion `SetDecimalPoint()` nicht beeinflusst, d.h. sowohl "." als auch "," werden automatisch bei der Eingabe von Dezimalzahlen via `ChartData()` als Dezimalpunkt erkannt.

SetThousandsSep(zeichen)

Die Funktion SetThousandsSep() eröffnet die Möglichkeit, Zahlen mit Tausender-Trennzeichen auszugeben. SetThousandsSep() ist eine "globale" Funktion, d.h. alle in einer Zeichung dargestellten Zahlen sind davon betroffen, unabhängig ob es sich dabei um Achsenbeschriftungen, Beschriftungen von Diagrammwerten, etc. handelt, auch unabhängig von eventuell vorhandenen Formatanweisungen. Beispiel:

```
OpenDrawing(200;120)
     SetThousandsSep("'")
     AddFrame(0;0;200;120)
     ChartData(10000 8000 5000 3000 2000)
     PieChart(label+oval)
     BorderStyle(all;none)
CloseDrawing()
```



Tausender-Trennzeichen sind bei der Eingabe von Zahlen nicht erlaubt — siehe Funktion ChartData() bzw. ChartDataRead() im Abschnitt *Daten*.

Index

Index