Was ist J?

J ist eine universell einsetzbare, array-orientierte Programmiersprache. Erfunden und entwickelt wurde J von Kenneth E. Iverson(Turing-Preisträger), implementiert von Roger Hui.

Joachim Hoffmann LIT-119

J ist ein Denkwerkzeug !

Die *Sprache* J wurde nach dem Vorbild menschlicher Sprachen entwickelt, und ist damit dem menschlichen Denken näher als herkömmliche Programmiersprachen, die in ihrer inneren Struktur die primitive, lineare Funktionsweise der Computer widerspiegeln.

Das Resultat ist eine am Computer ausführbare Sprache mit einer einfachen Grammatik und Syntax.

Aufgrund dieser einfachen Grundstruktur von J bleiben J-Programme kurz und überschaubar.

Schon mit wenig Übung beginnt man in J zu denken und in der Folge Probleme effizienter zu lösen.

J The Accessible Language

Bei J wurde besonderer Wert auf die Kommunikation mit anderer Software gelegt: J 32-Bit DLL, OLE2 Automation, ODBC Driver, DDE, ...

Die Idee hinter J

Iverson war in den 50-er Jahren an der Harward Universität mit der Aufarbeitung der ökonomischen Thesen von Wassily Leontief (Nobelpreis) am Computer beauftragt. Er erkannte bald, daß er mit den damals zur Verfügung stehenden Mitteln, nämlich der Maschinensprache des MARK I von Prof. Aiken, sehr lange damit beschäftigt sein würde. Das hat Iverson dazu veranlaßt, darüber nachzudenken, auf welche Art und Weise Mathematik und elektronische Datenverarbeitung möglichst sinnvoll und effizient „niedergeschrieben" werden könnte: Wie funktionieren mathematisch-technische Notationen? Wie und unter welchen Umständen sind sie entstanden? Warum gibt es für Addition und Subtraktion die Symbole + und -, aber keines für das Maximum von zwei Zahlen oder für das Lösen von Gleichungssystemen? Wie muß eine Sprache aufgebaut sein, damit man mit ihr effizient Probleme lösen kann?

Als erstes Resultat dieser Überlegungen entstand das Buch „A Programming Language" (Wiley&Sons, New York, 1962), in dem Iverson eine logisch in sich widerspruchsfreie technische Sprache beschreibt. Z. B. entfernt er alle historisch gewachsenen Syntaxregeln (wie z.B. die Hierarchie unter den Funktionen, Punkt-vor-Strich-Regel) und führt für alle grundlegenden Prozesse der Datenverarbeitung Symbole ein. Ein großer Teil seiner Ideen wurde Ende der 60-er durch IBM in der Programmiersprache APL verwirklicht.

Über die Jahre hat Iverson seine Ideen weiterentwickelt. Nach intensivem Studium der Funktionsweise von menschlichen Sprachen und in der Absicht funktionales Denken auf einfache Weise zu ermöglichen, hat Iverson seine Arbeit überdacht, Fehler analysiert, noch einmal von vorne begonnen. Das Resultat ist „J Introduction and Dictionary".

Welche Vorteile bringt Ihnen J?

Mit einer Zeile in J können Sie Probleme lösen, die in herkömmlichen Programmiersprachen (Pascal, C) hunderte Zeilen Code benötigen würden. (Dick Pountain in „The Joy of J", BYTE, September 1995).

J ist leicht zu lernen,

- weil die Syntaxregeln einfach, logisch einsichtig, und ohne Ausnahmen sind.
- weil man mit dem J-Interpreter sofort wie mit einem gigantischen Taschenrechner experimentieren kann. Schon ein Anfänger, der J nur "gebrochen spricht", ist in der Lage interessante Programme schreiben.

Obwohl J am Beginn wegen seiner Neuartigkeit trotzdem etwas Befremdung auslösen kann, rentiert es sich diese Hürde zu überwinden, weil man danach über ein mächtiges Denkwerkzeug verfügt, das es ermöglicht komplexe Probleme leichter gedanklich zu erfassen, und diese kurz und prägnant niederzuschreiben.

- Die Entwicklungszeiten werden minimal,
- die Programme bleiben übersichtlich und wartungsfreundlich,
- und die Produktivität steigt.

Dazu kommt noch der Spaßfaktor, wenn man Probleme schneller und effizienter als die Konkurrenz bewältigen kann.

J ist eine interpretierte Sprache:

- Anwendungen werden in kleine Module aufgeteilt, die sofort interaktiv getestet werden können, ohne ständig kompilieren zu müssen.
- Der Interpreter erledigt lästige technische Details - wie Datentyp-Deklaration oder Memory Allocation - und ermöglicht es dem Programmierer kann, sich ganz auf das eigentliche Problem zu konzentrieren.
- Sie haben mit J nicht die Geschwindigkeitseinbußen, die von anderen interpretierten Sprachen bekannt sind, weil in J jede Funktion auf ganze Arrays angewendet werden und somit die Notwendigkeit von Schleifen minimiert wird.

Wozu eignet sich J besonders:

- zum Beschreiben von Algorithmen:
  - bei komplexen analytischen Problemen
  - für das Manipulieren von Daten
  - im Unterricht
- beim schnellen Entwickeln von Anwendungen und Prototyping
- als Calculation-Server für Visual-Basic, Delphi, Excel, als komplettes System, z.B. von GUI-Programmierung bis zur maßgeschneiderten Datenbank

Wer verwendet J?

- Causeway Graphical Systems Ltd. entwickeln den Configuaration-Manager für SAP/R3.
- Motorola California setzt J bei der Entwicklung von Handschrifterkennungssoftware ein.
- Fax Focus Ltd., Aptos California verwaltet und analysiert mit J monatlich ca. 20.000 ein- und ausgehende Faxe, einschl. der dazugehörigen Telefonrechnung.

Auf Welchen Plattformen läuft J ?

J läuft auf allen populären Plattformen: DOS, WINDOWS.95.NT, Mac-Inthosh, OS/2, UNIX: LINUX, SUN, RS/6000, ... Die Kernspache J ist bezüglich der verschiedenen Hardware-Plattformen 1:1 kompatibel. Z.B. Sie können eine Anwendung auf einem PC entwickeln und testen, das Programm (=ASCII File) dann zu einem UNIX-Rechner schicken und dort ohne eine Änderung am Code ausführen lassen. (OLE-Automation gibt es unter UNIX natürlich noch nicht)

Die zentrale Plattform für J ist Windows95.NT mit:

- **OLE2 Automation** (JEXESERVER, JDLLSERVER, JCLIENT): Z.B: Sie entwickeln eine Benutzeroberfläche in Visual-Basic oder Delphi,

führen aber mittels OLE die anschließende Rechnung viel effizienter in J durch, und stellen die Ergebnisse in Excel dar.

- **32 Bit ODBC Treiber:** Das sind schnelle und vollständige Datenbanktreiber, die es Ihnen ermöglichen von J aus auf riesige von Datenmengen zuzugreifen.

- **DDE und DLL Support:** Unkomplizierte Kommunikation mit anderen Programmen über DynamicDataExchange und Einbindung vorhandener Lösungen als DynamicLinkLibraries.

- **Visual J:** Window Treiber zum GUI-Programmieren; komplettes GUI-Programming-Interface zum "FensterI-Programmieren" mit einem praktischen Visual Form Editor.

Seit wann wird J entwickelt?

- Begonnen 1989, basierend auf mehreren Jahrzehnten Forschung und Erfahrung von Ken Iverson & Co.

- Shareware-Ausgaben von 1990 bis 1993

- kommerzielles Produkt seit 1994

- Aktuelle Version: J 3.02 für Windows95/NT

J schafft Ihnen einen Wettbewerbsvorteil !

J ist leicht zu lernen,

- weil die Syntax-Regeln einfach, logisch einsichtig, und ohne Ausnahmen sind.

- weil man mit dem J-Interpreter sofort wie mit einem gigantischen Taschenrechner experimentieren kann.

Obwohl J am Beginn wegen seiner Neuartigkeit etwas Befremdung auslösen kann, rentiert es sich diese Hürde zu überwinden, weil man danach über ein mächtiges Denkwerkzeug verfügt, das es ermöglicht

- komplexe Probleme leichter gedanklich zu erfassen, und diese kurz und prägnant niederzuschreiben.

- Die Entwicklungszeiten werden minimal,

- die Programme übersichtlich und wartungsfreundlich,

- die Produktivität steigt. Mit J ist Prototyping kein Luxus mehr.

Zudem läuft J auf den meisten populären Hardwareplattformen (DOS, Windows, Macintosh, LINUX, Sparc, RS/6000) und J-Programme - Standard-ASCII-Dateien - sind *vollständig kompatibel*.

Wozu eignet sich J besonders:

- zum Beschreiben von Algorithmen:
  - bei komplexen analytischen Problemen
  - für das Manipulieren von Daten
  - beim schnellen Entwickeln von Anwendungen und Prototyping

- als Calculation-Server für Visual-Basic, Delphi, Excel, ...

- als komplettes System, z.B. von GUI-Programmierung bis zur maßgeschneiderten Datenbank

Was unterscheidet J von anderen Programmiersprachen?

- **Die Syntax ist einfach und konsistent.**
  - Zur Beschreibung der Sprache J wurden von Kenneth Iverson die Begriffe der englischen Grammatik herangezogen, weil diese die Grammatik von J besser beschreiben als die Ausdrücke, die normalerweise für Computerprogrammiersprachen oder in der Mathematik verwendet werden.
  - i. Eine Funktion (ein Programm) wird *Verb* genannt, weil es eine Aktion durchführt.
  - ii. Eine Einheit, die ein *Verb* modifiziert heißt *Adverb*.
  - iii. Eine Entität, die zwei Verben miteinander verbindet heißt *Konjunktion*
  - *iv.* Variablen, Zahlen, Arrays werden *Nomen* genannt.
  - Die Abarbeitung der Verben erfolgt von rechts nach links - ohne Hierarchie unter den Funktionen: Z.B. gibt es keine „Punkt vor Strich“-Regel.
  - Die Grammatik und Syntax von J ist auf nur 20 Seiten vollständig beschrieben.

- **Viele Funktionen manipulieren Daten.**

- **Viele Funktionen manipulieren Funktionen.**

J stellt Operatoren (*Adverbien* und *Konjunktionen*) zur Modifizierung und Kombination von Funktionen zur Verfügung.

Das „Insert“ Adverb „/“ ist ein typisches Beispiel:

Das Verb „+“ und das Adverb Insert „/“ bilden das modifizierte Verb „Summe einer Liste“,

```
        +/ 2 4 6
12
        2+4+6
12
```

/ das Produkt einer Liste, *<./* das Minimum einer Liste, ...

Oder das Hintereinanderausführen von Verben mit der Konjunktion „nach“ (@):

```
   (+&2)@(*&10) 3   NB. bedeutet:
32                  NB. (plus mit zwei) nach
                    NB. (multipliziere mit 10)
                    NB. & bindet das Substantiv 2
                    NB. an das Verb + um das neue Verb
                    NB. („plus 2) zu bilden.
```

- **Das stille oder implizite Programmieren:**

In J ist es möglich, Verben (Funktionen) ohne die explizite Erwähnung von Nomen (Variablen) zu einem neuen Verb zu kombinieren, deswegen „stilles“ oder „implizites“ Programmieren.

Dem Namen `sum` wird das Verb „Summe einer Liste“ zugewiesen (functional assignment).

```
        sum=. +/
```

Will man das arithmetische Mittel dieser Liste von Zahlen, schreibt man einfach:

```
        mittel=. sum  %  #
        mittel  2 4 6
4
        (sum 2 4 6) % (# 2 4 6)
4
```

In Worten: <Die Summe> dividiert durch <die Anzahl der Elemente in der Liste>.Man beachte bitte, daß bei der Definition von `mittel` weder eine Variable als Argument angeführt ist, noch eine Schleife verwendet wird. Das *Verb* `mittel` ist eine Abfolge von drei Verben oder eine Gabel, die wie folgt definiert ist: (f g h) y _ (f y) g (h y).

Die Anzahl der Basisregeln dieser Art in J ist minimal! (Zusätzlich zur Gabel ist nur noch der Haken definiert.)

- **Der Interpreter ist „sauber“ implementiert und leicht zu portieren.**
  - Es wird klar zwischen Sprache (Interpreter) und Bedienungsoberfläche (SessionManager) unterschieden.

- Das Design der Sprache und der Interpreter wurden komplett neu entwickelt.
- **Der Interpreter erledigt die lästigen technischen Details** - wie Datentyp-Deklaration oder Memory Allocation - und ermöglicht es dem Programmierer sich ganz auf das eigentliche Problem zu konzentrieren.
- Sie haben mit J nicht die Geschwindigkeitseinbußen, die von anderen interpretierten Sprachen bekannt sind, weil alle Funktionen auf ganze Arrays angewendet werden, und somit Schleifen minimiert werden.
- Sie können Ihre Anwendungen in kleine Module unterteilen und diese testen, ohne ständig Kompilieren zu müssen.

Was unterscheidet J von APL ?

- **J ist einfacher und konsistenter**
- **J führt die besten Ideen von APL fort,**
- **löst aber Probleme von APL:**
  - Spezieller Zeichensatz
  - Fehler im Design von APL
  - Isolation von anderer Software
  - Schwierigkeiten beim Hinzufügen von neuen Features
- **Die Entwicklungs-Umgebung in J ist Standard:**
  - Programme sind normale Text-Dateien (ASCII).
  - Jeder Texteditor kann zum Programmieren verwendet werden.
  - Jedes Versions-Kontrollsystem kann verwendet werden.
  - Leichter Zugang zu und von anderen Programmen.
- **Mächtige neue Fähigkeiten:**
  - Weiterentwickelte APL-Basisfunktionen
  - Rank von Funktionen
  - Funktionales Programmieren
  - Kontroll-Strukturen
  - Namensräume (Locales / Namespaces) um den Workspace ähnlich dem DOS-File-System in Unter-Workspaces einteilen zu können.

Seit wann wird J entwickelt?

- Begonnen 1989
- Shareware Ausgaben von 1990 bis 1993
  - Versionen 1 bis 7
- Ab 1994 kommerzielles Produkt:
  - J Release 2 Version 2.01
  - J Release 2 Version 2.06
- **aktuell J Release 3 für WIN95/NT**

Welche Hardware-Plattformen werden unterstützt?

- **Windows** 3.1, Windows NT, Windows95, OS/2
- **Macintosh**
- **UNIX**: Linux, SUN, RS/6000 - weitere auf Anfrage

Die Kernsprache von J ist zwischen allen Systemen 1:1 kompatibel.

Windows 95/NT

Die zentrale Entwicklungsplattform ist Windows NT und es gibt bereits ein Produkt das auf Windows 95 und NT läuft. Diese Plattformen sind sehr robuste Betriebsysteme, die eine weitaus größere Funktionalität bieten als Windows 3.1. Der J Interpreter ist eine vollständige 32-Bit Implementation mit beachtlichen Geschwindigkeitsverbesserungen, unter anderem durch die Verwendung von 32-Bit ODBC und 32-Bit DLLs.

Leistungssteigerung

Die Leistung von Version 3 (J3) wurde dramatisch verbessert. J3 ist mindesten 5 mal schneller als die vorige Version und übertrifft jetzt die meisten array-orientierten Programmiersprachen.

J3 beinhaltet auch integrierten Funktions-Rank, sodaß die Anwendung von Funktionen mit Rank nun sehr effizient ist.

OLE Automation

Die aufregendste Neuerung dieser Version ist mit Sicherheit die OLE Automation in J3 für Windows 95/NT, das es J erlaubt direkt mit anderen Anwendungen zu kommunizieren.

J ist ein wunderbares (Denk-) Werkzeug um Algorithmen auszudrücken. J eignet sich daher ideal für Daten- verarbeitende Anwendungen ("data processing"), wogegen andere herkömmliche Programmiersprachen sich hauptsächlich auf die Herstellung aufwendiger grafischer Benutzerschnittstellen (GUI) konzentrieren. J OLE ermöglicht es Ihnen jetzt, die Vorteile aus beiden Welten zu nutzen.

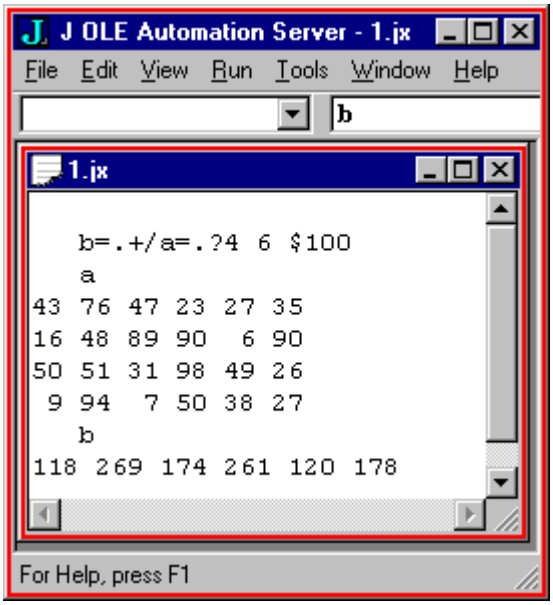
Sie können J auf zwei Arten verwenden:

☺ Für Anwendungen, die nur Standard Windows GUI-Funktionen benötigen, können Sie J als komplettes Entwicklungssystem verwenden.

☺ Im Gegensatz dazu ist es nun möglich, eine grafische Benutzerschnittstelle in Visual Basic oder Delphi zu entwickeln und J direkt von diesen Plattformen aus als Berechnungs-Server zu benutzen - mittels OLE Automation ist das ein leichtes.

In beiden Fällen bietet Ihnen J mächtige Möglichkeiten zur Datenverarbeitung, die von keiner anderen Programmiersprache auch nur annähernd erreicht werden.

Zusätzlich zu OLE können Sie auch DDE benutzen um mit J zu kommunizieren. DDE ist aber eher fragil und weniger effizient. Im Gegensatz dazu ist OLE die bevorzugte Methode um unter Windows 95/NT zwischen Programmen zu kommunizieren, weil es einfach, robust und leistungsfähig ist. Mit J Release 3 werden zwei OLE Automation Server angeboten:



JEXE-Server ist ein vollständiges J Entwicklungssystem, das hauptsächlich während der Entwicklung von Anwendungen eingesetzt wird. JEXE-Server hat dieselbe Funktionalität wie J.EXE, das das normale J

Entwicklungssystem darstellt, und verschafft Ihnen Zugang sowohl zu den J Client- als auch zu den J Serverumgebungen.

JDLLServer ist nur der J Interpreter und wird als "in-process" Server eingesetzt. JDLLServer eignet sich ideal als schnelle "Rechenmaschine" für Runtime Anwendungen. JDLLServer ist als gewöhnliches 32-Bit DLL implementiert und kann daher von jedem anderen Programm, das fähig ist 32-Bit DLLs aufzurufen, verwendet werden.

Visual J

Der Window-Treiber wurde konsequent weiterentwickelt und mit einem *visuellen FormEditor* ausgestattet, mit dem Sie Ihre "Fenster!" interaktiv und unkompliziert erstellen können.

Der Window-Treiber für Windows 95 unterstützt alle neuen GUI-Objekte wie z.B.: Progress Bar, Richedit, Spin, Tabs und Trackbar.

Runtime Systeme

Runtime Systeme sind ab sofort kostenlos erhältlich, sowohl als komplette Systeme einschließlich des Window-Treibers, sowie als 32-Bit DLL mit der Kernsprache alleine. Runtime Systeme haben keinen SessionManager, sind aber sonst der Professional Edition gleichwertig. Die Runtime Interpreter führen die Script-Datei in ihrer Kommandozeile aus. Diese Script-Datei kann ein ASCII-Text (*.js Datei) oder ein kodierter Text (*.jr Datei) sein. Jedes J für Windows System kann Runtime Anwendungen erzeugen, die ohne Einschränkungen verteilt werden dürfen.

Neue Verben

Die J Kernsprache wird ständig verbessert. Seit J2.06 wurden folgende neue Funktionen entwickelt:

☺ Numerisch exakte Integer-Arithmetik, z.B. "2 hoch 101"

```
  2^101x
2535301200456458802993406410752x
```

☺ **OCX** Unterstützung

☺ Die Level Konjunktion, um Funktionen in beliebiger Tiefe von verschachtelten Arrays anzuwenden.

☺ Das Level Verb, um die Tiefe eines verschachtelten Arrays (boxed noun) zu ermitteln.

☺ Die Sortier-Funktionen wurden auf alle Typen von Arrays (somit auch auf verschachtelte) verallgemeinert.

☺ Neue Debug Funktionen

☺ Ein Latenter Ausdruck, der bei jedem Error aufgerufen wird.

Bücher

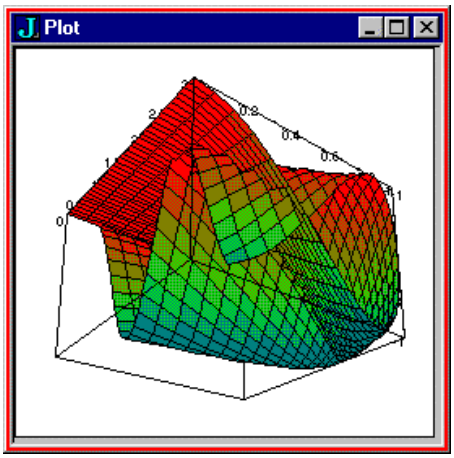
J Phrases ist ein neues Buch, das viele nützliche Hilfsverben und Phrasen enthält, die für den Anfänger zum Lernen genauso nützlich sind wie für den Profi zum Nachschlagen. J Phrases enthält Verben für eine Vielfalt von Themen, unter anderem: ... indexing and replacing, searching and selecting, inverse and duality, statistics, matrix algebra, ...

„J Phrases", „Introduction and Dictionary" und das „UserManual" sind als Windows-Hilfe-Datei und als Bücher im J-System enthalten.

Introduction and Dictioanry	**ATS 400,-**
UserManual	**ATS 400,-**
Concrete Math Companion	**ATS 400,-**
Programming in J	**ATS 250,-**
Arithmetic	**ATS 250,-**
Calculus	**ATS 250,-**
Fractals Visualization and J	**ATS 500,-**
J Conference Proceedings 96	**ATS 400,-**
J Primer	**ATS 400,-**

Praktische Utilities

Mit der *Plot* Package können die verschiedensten Arten von Diagrammen erzeugt werden. Oder das *Laboratory* zum Erlernen und Debuggen von *stillen* Definitionen.



J User Conference

Die erste J User Konferenz hat Ende Juni 96 in Toronto/Kanada stattgefunden.

Es wurden dort kommerzielle J-Anwendungen präsentiert: z.B. Motorola Kalifornien setzt J für Handschrifterkennungs-Software ein, oder Fax Focus Ltd. analysiert und verwaltet pro Monat ca. 20.000 ein- und ausgehende Faxe mit J.

Ein weiterer Schwerpunkt wurde auf die Kommunikation von J mit anderen Programmen mittels OLE Automation und ODBC gelegt. Z.B. demonstrierte J. D. Baker (Ministry of Health, Kingston, Ontario) die Verwendung von J im Zusammenspiel mit FoxPro und Delphi, und Causeway Graphical Systems präsentierten den neuen Configuration-Manager für SAP/R3 ...

J vereinfacht Programme

Denken Sie gerne in mehrfach verschachtelten Schleifen? Macht es Ihnen Spaß nur in IF THEN ELSE oder WHILE DO END Strukturen zu programmieren? ***Nein!***

Dann sollten Sie einmal einen Blick auf J werfen. Zur Veranschaulichung ein nicht ganz einfaches Programmbeispiel:

Wie würden Sie überflüssige Leerzeichen aus einem Text löschen?

```
'Heinz     Kurt   Xenia      ' --> 'Heinz Kurt Xenia'
```

J-typische Lösungen benötigen dafür nur eine Zeile!

Mittels Hilfsverben wird das Problem in kleine Untereinheiten unterteilt:

```
lrb=. (' '&,)@(,&' ')     NB. left right blank
fdb=. '  '&E.             NB. find double blanks
DEB=. (#~  -.@fdb)&.lrb   NB. Delete Extraneous Blanks
```

Es ist jetzt nicht notwendig, daß Sie diese Einzeiler verstehen, alleine der Vergleich der Länge mit dem folgenden Beispiel ist hier von Bedeutung.

In einer herkömmlichen Programmiersprache, wie z.B. Basic, Pascal oder C, würde das umständlich mit Schleifen- und IF-THEN-Strukturen ungefähr folgendermaßen programmiert werden müssen:

Wieviel kostet J?

J3.02 Win95.NT, Win3.1-Professional	**ATS 9.000,-**
J3.02 Win95.NT, Win3.1, Macintosh-Standard	**ATS 4.500,-**
J3.02 DOS, LINUX, SUN, RS/6000	**ATS 3.000,-**

J Web Page

Unsere neue Web Page unter **http://www.jsoftware.com** enthält die Manuals der aktuellen J Version, J Runtime Executables, J FreeWare, Utilitys, knifflige Rätsel, ...

J Österreich, Iverson Software Inc., Austria

✉ *Joachim Hoffmann, Münzgrabenstr. 68, A-8010 Graz*
☎ *0316 81 45 29, fax: 0316 81 66 83*
E-✉ *JoHo@magnet.at*
WEB *http://www.jsoftware.com*

Programmbeispiele in J

Im Folgenden die Aufzeichnung einer konkreten J-Sitzung: Am linken Rand steht die Antwort des Systems.

```
NB. --- Namen alphabetisch sortieren ---------------------------------

   words  =. ;:             NB. Woerter bilden / funktioanle Zuweisung
   sort   =. /: ~           NB. Sortieren
   under  =. &.             NB. Konjunktion: f&.g <=> (inv g) f g
   bywords=. under words    NB. etwas Wort-weise tun / adverb

   text=. 'Heinz Xenia Kurt Leopold'   NB. Liste mit Buchstaben

   sort bywords text        NB. sortiere Worte im Text
Heinz Kurt Leopold Xenia


   reverse=. |.             NB. umdrehen
   reverse bywords text     NB. Reihenfolge der Worte im Text umdrehen
Leopold Kurt Xenia Heinz
   text                     NB. urspruenglicher Text zum Vergleich
Heinz Xenia Kurt Leopold
   reverse text             NB. Text umdrehen
dlopoeL truK aineX znieH
```

```
NB. --------- Das Arithmetische Mittel ---------------------------------

   +/ 1 2 3 4 5           NB. Summe ueber Liste
15
                          NB. das entspricht der Anweisung:
  1+2+3+4+5
15

  */1 2 3 4 5             NB. Produkt ueber Liste, %/ , >./ Maximum
120

   sum=. +/               NB. funktionale Zuweisung
   sum 1 2 3 4 5          NB. +/ entspricht dem Summenzeichen Sigma
15
   count=. #              NB. Anzahl der Elemente in einem Array
   mean =. sum % count    NB. Das arithmetische Mittel
   mean 1 2 3 4 5         NB. ! das war soeben FUNKTIONALES PROGRAMMIEREN
3                         NB. ! = das Kombinieren von Funktionen (Verben)
                          NB. ! zu einem neuen Verb

   (sum 1 2 3 4 5) % (# 1 2 3 4 5)
3
   under=. &.             NB. Konjunktion zum Verbinden von Verben:
                          NB.      g &. f <=> (f invers) g f
   hmean=. mean under %   NB. Das Harmonische Mittel:
                          NB.      Der Kehrwert des Mittels der Kehrwerte

   hmean 1 2 3 4 5
2.18978

   % mean % 1 2 3 4 5
2.18978
```

```
NB. Gruppiere Verkaufszahlen nach Artikelnummer

   items=.  1  2  2  3  2  3  1  1  2
   sales=. 20 25 30 30 15 20 10 15 15
   box  =. <   [.   sum=. +/
   items box/. sales
+--------+-----------+-----+
|20 10 15|25 30 15 15|30 20|
+--------+-----------+-----+
   items sum/. sales
45 85 50
```

```
NB. ---------- hexadezimal Rechnen ---------------------------------

   h2d=. (16&#.)@('0123456789ABCDEF'&i.)  NB. hexadezimal nach dezimal
   under=. &.                             NB. f&.g <=> (inv g) f g
   HEX=. under h2d                        NB. HEX adverb

   'A' + HEX '1'
B
   'B' - HEX '1'
A
   'A' * HEX '5'
32
   'A' % HEX '5'
2
```

```
NB.  Fakultaet   1.) Schleifenprogramm
NB.               2.) Direkt Definiert
NB.                3.) Basisfunktion !

factorial=. 3 : 0             NB. Als Schleifen Programm
a=. 1                         NB. y. ist das rechte Argument
 while. y. > 1                NB.        des Verbs factorial
  do. a =. a  * y.
      y.=. y. - 1
 end.
a                             NB. a ist das Ergebnis
)


fac=. 1:`(] * fac@<:)  @. *   NB. direkt und rekursiv definiert

   factorial 9
362880
   fac 9
362880
   !9
362880
```

```
NB. --- Ueberfluessige Leerzeichen aus einem Text loeschen - 2 Versionen

lrb  =. (' '&,)@(,&' ')       NB. left right blank
fdb  =. '  '&E.               NB. find double blanks
DEB =. (#~ -.@fdb)&.lrb       NB. Delete Extraneous Blanks

   DEB '      The     Joy     of    J      '
The Joy of J
```

```
NB. ---  Das gleiche im Schleifenstil - zur Abschreckung ! --------------

DEB_LOOP=. 3 : 0
NB. Delete Extraneous Blanks - Looping Version
and=. *.  [. not=. -.      NB. functional assignment of logical and
blk=. ' ' [  res=. ''      NB. blank / result - empty
inx=. 0                    NB. index
nbr=. (# y.)-2             NB. number of loops

 while. inx <: nbr do.                    NB. compare adjacent elements
  if. not (blk=inx{y.) and (blk=(inx+1){y.)NB. if both are not blank then
    do. res=. res, inx { y.       NB. the first is part of the result
   end.
  inx=. inx + 1                   NB. counting index
 end.                             NB.

 if. 0 ~: # res do.               NB. check if result is empty

  if.   blk = {. res              NB. if first element is blank
   do.  res=. }. res              NB. delete it
  end.

  if.  blk = {: res               NB. if last element is blank
   do. res=. }: res               NB. delete it
  end.
 end.
res                               NB. result
)
```

Wie verschiedene Programmierer ihre Fahrräder bauen:

(Nachtrag zu PCNEWS edit-49, Seite 56)

J'uggler bauen etwas rundes und rollen damit überall hin.