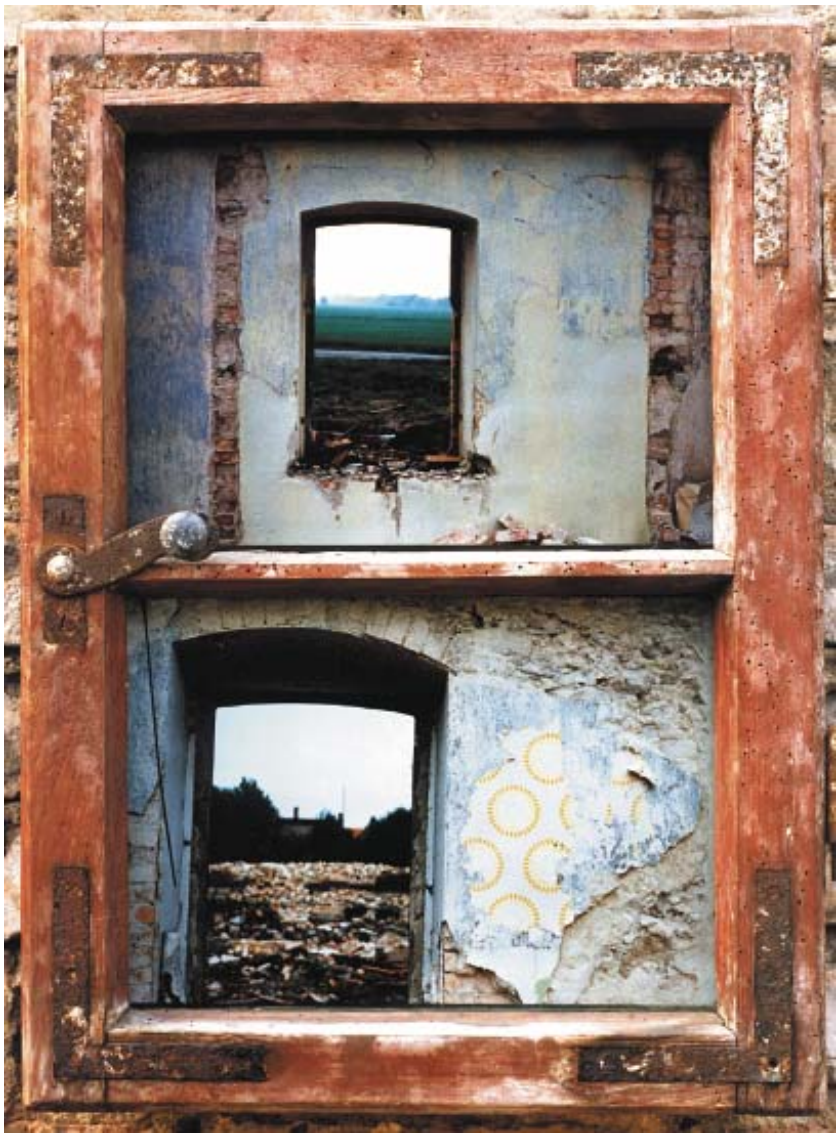
APL-Journal

A Programming Language

2/2000

APL-Germany e.V.
RHOMBOS-VERLAG
ISSN - 1438-4531 Jahrgang 19 Nr. 2 2000

In dieser Ausgabe:



Im Blickpunkt

Brigitte Denck

Fenster, Skulpturen, Installationen

Inhaltsverzeichnis

4 Programmiersprachen: APL In the New Millennium

Since IBM's APL\360 became available in 1966, many dialects have been developed, and competition has led to emphasis on their differences, an emphasis reflected in their distinctive names: APL\1130, APL\360, APLSV, APL2, SHARP APL, Nial, Dyalog APL, A, APL2000, J, K, and others. In this article Kenneth E. Iverson will review developments in the APL dialects, emphasizing similarities and the ways in which competing ideas have been, and could be, shared and adapted to competing systems. His hope is to encourage the relatively small APL family to mute their differences, and present a more united face to the programming world.

7 APL2-Funktionen: Neuer Service-Level

Für die IBM APL2-Workstation-Produkte ist seit Dezember 2000 zusätzlich zu den üblichen Fixes für gemeldete Fehler ein neuer Service Level verfügbar (CSD7 für das Windows-Produkt, CSD18 für OS/2, AIX/6000 und Sun Solaris). Für APL2/390 V2.2 sind entsprechende Ergänzungen via PTF ebenfalls erhältlich. Im folgenden Beitrag werden die wichtigsten Neuerungen vorgestellt, zu denen beispielsweise verbesserte Internet-Werkzeuge, eine optimierte ODBC-Unterstützung und neue Windows Standardsteuerelemente zählen. Neben einer Übersicht zu wichtigen Funktionsmerkmalen von IBM APL2 für Windows werden die Eigenschaften der Prozessoren und mitgelieferten Hilfsprozessoren für die Schnittstellen zu den System-Services aufgelistet und die Möglichkeiten der APL2 Runtime Environment for Windows skizziert.

12 Berechnen des Effektivzinses: PAngV oder Newton Revisited

Mit dem Newton-Verfahren kann man für eine gleichbleibende Zahlungsperiode den effektiven Jahreszins ausrechnen, den auch die Hausbank angibt. Wählt man aber andere (unterjährige) Zeiträume, so stimmen die Ergebnisse nicht mehr überein. Dies liegt daran, dass die deutschen Banken bislang nach Vorschriften rechnen, die durch die Preisangabeverordnung (PAngV) gesetzlich vorgegeben sind. Diese Verordnung soll nach dem Willen der EU durch Regeln der ISMA (International Securities Market Association) ersetzt werden und die Definition des Effektivzinses in der EU vereinheitlichen. In diesem Beitrag werden Vorgehensweisen aufgezeigt, wie der Newton Algorithmus auf unregelmässige Zahlungsperioden angewandt werden kann.

Impressum APL - Journal ISSN 1438-4531

Herausgeber: Dipl.-Math. Dieter Lattermann, Walldorf

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantwortlich)
Technische Universität Berlin
Franklinstr. 28
D-10587 Berlin
Tel. (030) 314 24392
Fax (030) 314 25901
Email: barg@cs.tu-berlin.de

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Verlag: RHOMBOS-VERLAG, Berlin
Kurfürstenstr. 17
D-1785 Berlin
Tel. (030) 261 9461
Fax (030) 261 6300
Email: verlag@rhombos.de,
Web: www.rhombos.de

Gestaltung: RHOMBOS-VERLAG, Berlin

Druck: Kopierfabrik, Eberswalde

Homepage des APL-Journals: http://www.rhombos.de/apljourn.htm
FTP-Server zum Datei-Upload: ftp.cs.tu-berlin.de/pub/lang/apl/stat/incoming.

Wie immer freuen wir uns auf Ihr Feedback zu diesem Heft. Schreiben Sie uns an barg@cs.tu-berlin.de, oder faxen Sie an 030 - 314 25901.

Technische Hinweise für Autorinnen und Autoren

Manuskripte werden von der Redaktion entgegengenommen. Sie müssen im Besitz der Nutzungsrechte für die von Ihnen eingereichten Texte, Fotos und Zeichnungen sein. Die Zustimmung zum Abdruck wird vorausgesetzt.

Manuskripte und Beiträge aller Art werden in jeder Form entgegengenommen. Es werden jedoch Beiträge in gespeicherter Form bevorzugt. Bitte speichern Sie Ihren Text im Format Ihrer Textverarbeitung und zusätzlich als RTF-Datei (Rich Text Format). Sie brauchen sich nicht die Mühe machen, Ihren Text aufwendig zu formatieren. Auszeichnungen im Text sollten sich auf „Kursiv“ beschränken. Ihre Datei schicken Sie der Redaktion bitte auf einem Datenträger (Diskette oder CD-ROM). Alternativ hierzu können Sie der Redaktion Ihre Daten auch per Email (barg@cs.tu-berlin.de) oder auch zum FTP-Server der Technischen Universität Berlin schicken. Die Adresse lautet: ftp.cs.tu-berlin.de/pub/lang/apl/stat/incoming.

Auch ein komfortables FTP-Transferprogramm für Windows gibt es beim ftp-Server des Fachbereichs Informatik der TUB unter: ftp.cs.tu-berlin.de/pub/lang/apl/bin/

Wenn Sie Ihr Manuskript einreichen, sollten Sie zusätzlich eine Gliederung und Zusammenfassung Ihrer Arbeit vorlegen. In der Zusammenfassung sollte in wenigen Sätzen die wichtigste Aussage Ihres Textes wiedergegeben sein.

Grafiken

Bitte senden Sie uns einen sauberen Ausdruck der Graphiken auf rein-weißem Papier. Linien (keine Haarlinien!) in den Graphiken sollten Schriften so groß gewählt sein, dass sie auch bei (stark) verkleinerter Wiedergabe im Heft noch zu erkennen sind; auf der Rückseite vermerken Sie bitte Ihren Namen und die Nummer der Abbildung; Zusätzlich wäre es schön, wenn Sie uns die mit einem gängigen Vektor-Grafikprogramm erstellten Zeichnungen und Grafiken auch als Datei liefern. Grafikdateien (Strichzeichnungen) sollten möglichst im Austauschformat EPS (skalierbar) mit 300 bis 600 dpi Auflösung in schwarzweiß bzw. Graustufe abgespeichert werden. Nach Rücksprache können sie zusätzlich Ihre Grafiken in Ihrem Original-Dateiformat liefern oder in ihr Textdokument einbetten.

Digitale Fotos benötigen wir im TIF-Format mit einer maximalen Auflösung von 250 dpi. Nicht geeignet sind Abbildungen und Fotos, wenn Sie aus dem Internet stammen beziehungsweise im Original als Bildschirmdarstellung (Auflösung 75 - 150 dpi) vorliegen. Ebenfalls nicht geeignet sind Grafiken, die mit einem pixelorientierten Zeichenprogramm erstellt wurden.

Editorial

Walldorf, 21. Februar 2001

Liebe APL-Freunde,

wieder einmal können wir Ihnen ein *APL Journal* überreichen. Es war nicht einfach, genügend Material zusammen zu bekommen, aber dank der Überredungskünste unseres Editors gelang es doch, eine weitere Ausgabe zu füllen. Den Autoren sei herzlich gedankt.

Da Sie erfreulicher Weise noch immer Mitglied von *APL Germany e.V.* sind, muss ich annehmen, dass *APL* in Ihrem Berufsleben oder zumindest als Hobby eine wichtige Rolle spielt. Zögern Sie deshalb nicht, uns davon zu berichten.

Wie schon im vorhergehenden *Journal* angekündigt, werden wir unsere nächste Tagung in Bingen abhalten, am 27. April 2001. Eine detaillierte Einladung werden Sie noch erhalten. Ich möchte noch einmal daran erinnern, dass Vorstandswahlen anstehen. Kandidaturen für ehrenvolle Aufgaben im Vorstand werden dringend gesucht. Insbesondere wird das Amt des Schatzmeisters neu zu besetzen sein, da Gert Osterburg nicht mehr zur Verfügung stehen wird.

In diesem Jahr wird die internationale APL-Konferenz unter dem Motto *APL2001: An Arrays Odyssey* vom 25.-28. Juni an der Yale University in New Haven, Connecticut, USA, stattfinden. Bitte beachten Sie die beigefügte Einladung.

Ich wünsche Ihnen eine interessante Lektüre und grüße Sie herzlich,

Ihr

Dieter Lattermann

Kenneth E. Iverson

APL In the New Millennium

Since IBM's APL\360 became available in 1966, many dialects have been developed, and competition has led to emphasis on their differences, an emphasis reflected in their distinctive names: APL\1130, APL\360, APLSV, APL2, SHARP APL, Nial, Dyalog APL, A, APL2000, J, K, and others.

Although natural to healthy competition, the emphasis on differences has discouraged the sharing of ideas, and still tends to blind programmers to the ease of moving between dialects, an ease not shared by programmers unschooled in the core ideas of APL.

As emphasized in [1], these core ideas were:

- **The adoption from Tensor Analysis of a systematic treat ment of arrays, in which every entity is an array, and different *ranks* lead to *scalars*, *vectors* (or *lists*), *matrices* (or *tables*), and *higher-dimensional arrays* (or *reports*).**
- **Operators (in the sense introduced by Heaviside [2]), which apply to functions to produce related functions.**

In this paper I will review developments in the APL dialects, emphasizing similarities and the ways in which competing ideas have been, and could be, shared and adapted to competing systems. My hope is to encourage the relatively small APL family to mute their differences, and present a more united face to the programming world.

Reprinted from *APL Quote Quad* (Vol. 30, No. 3) with permission

Alphabets

Although the particular alphabet, or even the font used, is a most superficial aspect of a language, it can make a dramatic assault on a beginning reader - as anyone who first met German in the Gothic font can testify. First encounters with the unfamiliar alphabet of the earliest APL has certainly discouraged many, in spite of its highly-mnemonic character.

At the time of its design there was no adopted standard, and it seemed reasonable to exploit the newly available IBM Selectric typewriter (with its easily-changed *typeball*) to design our own alphabet, and to use the backspace ability of the typewriter to produce composite (*overstruck*) characters.

The APL community was too small to influence the design of the now widely-used ASCII alphabet, and our use of special characters led to a series of unforeseen difficulties that have significantly inhibited the use of APL:

- When the "glass terminal" provided by the cathode ray tube supplanted the typewriter, it was incapable of backspacing to provide the composite characters of APL.
- APL characters were not provided by early printers, and there was a considerable delay before specialized alphabets could be downloaded to them.

Such difficulties have led some dialects (such as Nial) to adopt ordinary names as *reserved words*, an approach that the special characters had allowed us to avoid. Use of the internet also poses problems, because APL characters are not generally handled by browsers.

J and K use only the ASCII alphabet, but yet avoid the use of reserved words. K uses mainly single-character non-alphabetics, and J supplements these by a scheme that uses a suffixed period or colon. For example, `<` denotes *less-than*, `<.` denotes *lesser-of* (minimum), and `=:` denotes assignment.

It would, of course, be impractical for any APL system to switch to a rival alphabet, and this discussion is meant only to suggest supplements based on rival ideas. For example:

1. Many ASCII symbols (such as `@`, `&`, `^`, and `%`) go unused in most APL systems, and could be used in various ways:
   - One or two might be used as suffixes, as in `<@` or *`nub`*`@` as names for primitives.
   - The `*` adopted for power in APL\360 has since become universally recognized as the symbol for multiplication, and might better be replaced by the `^`, as first proposed by De Morgan in mathematics, and since adopted in some non-APL languages.

2. The percent symbol (which signifies "divided by 100") has been adopted for division by some non-APL systems, and could be more widely adopted in APL. After all, the APL symbol for division is no longer familiar in math.

Number representation

APL\360 introduced the useful representation of a negative number distinct from the negation function (-) that might produce it. This has been continued in all APLs in forms that vary from the special overbar symbol used in APL\360. Constrained to ASCII symbols, J uses the underbar, and K uses juxtaposition: `_3` for a negative number, as contrasted with `- 3` for negation.

APL systems use the exponential notation adopted from Fortran, some using the uppercase E, and some the lowercase. The notation has been extended to other kinds of numbers, as in `2d45` for a complex number in polar representation (APL2); `3j4` for a complex number (SHARP APL); and `2r3` for a rational number (J).

1 Iverson, K. E., A Personal View of APL, *IBM Systems Journal*, Vol 30, No. 4, 1991

2 Heaviside, Oliver, See the 1971 Chelsea Edition of Heaviside's *Electromagnetic Theory* and the article by P. Nahin in the June 1990 issue of *Scientific American*

Grammar

The grammar (parsing rules) of APL\360 were simple and relatively uniform, with no precedence among functions, but with operators given precedence over functions.

In particular, the relative positions of functions and arguments were fixed; for example, factorial n was written as !n, *not* n!. There are, however a few characteristics that merit comment.

Name assignment

A left arrow was used to assign names to arrays of numbers and characters, but a quite different mechanism was used for assigning names to functions, and there was no provision for defining operators.

In APL2, operator definition was introduced by extending the system for function definition to allow further parameters. Most systems have adopted this scheme, but J uses a single copula (the =: mentioned earlier) for all three cases, and Dyalog APL uses it for two.

Valence

APL\360 systematically adopted the double use of symbols from the scheme suggested by subtraction (3-2) and negation (-2) in mathematics, calling the two cases *dyadic* and *monadic*. For example, !n denotes the factorial, and m!n denotes the related binomial coefficients.

Most systems (with the exception of Nial) continued this *ambivalent* use of primitives, but did not extend it to the *derived* functions produced by operators.

In J, *all* functions are ambivalent. For example, `+/ y` denotes sum over *y*, and `x +/ y` denotes the addition table; that is, the plus-outer-product denoted by `x ∘.+ y` in APL\360 and APL2.

Indexing

Because of the need for multiple index arguments for a multidimensional array, APL\360 adapted from Fortran the notation `A[I;J;K]`, departing from the normal form for a dyadic function. In particular, it was not possible to assign a name to the complete index argument.

The introduction of *nested arrays* in APL2 made possible the treatment of a set of indices such as `I;J;K` as a single entity. However, many APL systems did not fully exploit this to rationalize indexing.

APL\360 introduced a further anomaly in indexing by providing either 1-origin or 0-origin indexing, set originally by a *system command* of the form `)ORIGIN`, and later by a *system variable*, `⎕IO`.

This choice of index origin has been maintained in most systems, but J is restricted to 0-origin. Since indexing in J is a normal dyadic function, an operator can be used to modify it, as well as the related index generator analogous to the iota of APL\360.

The restriction to 0-origin has simplified the introduction of *negative indexing*, with indices for n items running from negative n to n-1.

The *indexed assignment* `A[I;J]←M` is a further convenient (though grammatically anomalous) scheme introduced in APL\ 360. It is maintained in most systems, although the three essential arguments can be handled by an *amend* operator, such as the `}` used in J in the form:

```
M ind } A.
```

Terminology

Coming from a common background in math, we naturally adopted mathematical terminology in APL\360, in spite of the facts that:

- The term *operator* (used in the sense of Heaviside) conflicts with its common use in elementary mathematics as a synonym for function.
- The term *variable* used for a name assigned to a quantity suggests that its meaning might *vary* due to possible reassignment. But the same possibility exists for defined functions, and the terms *variable* and *function* do not aduately reflect the possible cases.

Terms from English grammar can provide the necessary distincions, using the close analogy between *function* and *verb* as denoting *actions*, together with the *nouns* and (variable) *pronouns* on which they act.

Moreover, *adverbs* are analogous to *operators* (such as /) that modify a single verb, and *conjunctions* (such as the copulative conjunction *and* in the phrase "run and hide") are analogous to operators (such as the inner-product) that apply to two verbs.

Finally, the familiar English terms *list*, *table*, and *report* are more commonly understood (and are fully as accurate as) the terms *vector*, *matrix*, and *higher-dimensional array*.

Opportunities

Most APL systems share unexploited cases that may be introduced without conflict. We will illustrate these opportunities by three classes.

Complex arguments

Although complex numbers serve primarily in mathematical expressions, their two parts (real and imaginary) can be convenient in functions that require the specification of two independent parameters.

For example, a format function *F* might be defined so that `12j3 F x` formats *x* with a width of 12 spaces and with 3 digits following the decimal point. A list of such complex arguments could be used to specify each column of the format of a table *x*.

Vector arguments

In APL\360, the expression `⍳5` produced a list of five successive integers, but the domain of iota did not include a vector argument. A useful non-conflicting extension could be defined to apply to a list of n integers to produce an array of successive integers of rank n (as in J) or a nested array of indices of an array of the same rank (as in Dyalog APL).

Trains

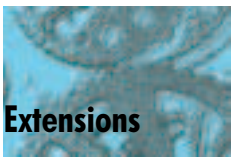
In calculus, the expression *f+g* is sometimes used to define a function equivalent to the sum of the functions *f* and *g*, and *f×g* is used for their product.

Moreover, such a train of three functions is treated as an error in most APL systems, and could therefore be introduced without conflict.

Any three functions may be used. For example `+/%#` (the sum divided by the number of elements) defines the *mean* function, and `f,g` defines a function that catenates results, as in `+/,*/`.

More generally, a train of any odd number of functions defines a function, the last three defining a function as stated above, and this function entering into a similar definition with the remaining train.

For example, the identity function followed by - (subtraction) followed by the preceding three-element train for the mean defines the function "center on the mean."



Extensions

Functions and operators new to any system can of course be adapted from other systems without conflict. In early systems the choice of symbols posed a problem, soon solved in a general way by the adoption of a class of "quad names"; alphabetic names preceded by the quad character. This solution appears to continue in systems that adhere to the special APL character set.

We will now discuss a few of the many functions and operators that are candidates for adoption.

GCD and LCM

The logical *or* and *and* could be construed as special cases of *maximum* and *minimum* (when restricted to the Boolean domain 0 and 1), or as special cases of *greatest common divisor* (GCD) and *least common multiple* (LCM).

E. E. McDonnell noted that only the latter functions possessed the same identity elements as *or* and *and*, and ensured that the logical functions were extended to GCD and LCM in the SHARP APL system. The same extension could be made without conflict in any APL system.

Repeatable random numbers

In debugging a program it is often useful to generate "random" arguments in a repeatable manner. This can be done by resetting the random seed. It is more convenient to provide a random number generator (denoted, perhaps, by ?) that resets the seed on each use.

Variants

APLSV used a *system variable* to specify the comparison tolerance to be used in relations such as `<` and `=`. Such control can be made more convenient by a variant operator, as in `= VAR 0` for exact comparison.

A more interesting opportunity for variants occurs in the case of the *rising* and *falling* factorial functions, defined by `*/x+s*` i.n, for `s` assigned the values `1` and `_1`, respectively. Moreover, a zero value for s gives the function `x` to the power `n`, and these functions (including the useful case of non-integer values for `s`) can all be treated as variants of the power function.

Ties

The sum `a+b+c+d+e` can be written as the reduction `+/a,b,c,d,e`. Moreover, the continued fraction `a+b/c+d/e` might lend itself to a similar reduction, except that it requires the *two* functions of addition and division.

Such a pair of functions (or any number) can be provided by a TIE operator, to be used in the form:

```
+TIE% / a,b,c,d,e.
```

The result of a tie can be used in other ways, as with a *case* operator, in which

f TIE g TIE h CASE i uses the index produced by the function i to select one of the functions for execution. In particular, the tie of two functions can be used to make a recursive definition.

Universal sorting

A strict ranking can be imposed on *all* arrays (including characters, real and complex numbers, and open and nested arrays of any rank) so that sorting can be applied to *anything*. An example of such ranking is provided in J, and could be adapted to any system.

The workspace

When first proposed by Adin Falkoff for APL\360, the (32K) workspace provided a convenient and efficient organization of memory. However, the fixed size, and other characteristics of the workspace, have come to inhibit the use of APL.

In particular, the workspace organization did not facilitate the exploitation of the memory management offered by later machines and operating systems. Arthur Whitney made the first step in employing these facilities in his A system, and used text (*script*) files rather than workspaces.

The advantages of text files are beginning to be recognized. In an item on page 55 of the April issue of *Vector* (Vol. 16, No. 4), Ansii Seppala is quoted as follows: *"the more I can work with text files, the easier it is - I am no longer a fan of the APL workspace."*

Acknowledgment

I am indebted to Chris Burke for many helpful comments, particularly for his suggestion to discuss the matter of the workspace versus script files.

Contact:

Kenneth E. Iverson can be reached via e-mail at "kei@interlog.com" or by phone at 1-416-924-0007.

Axel Güth

Neuer Service-Level

Die IBM APL2-Produkte wurden um wichtige Funktionen erweitert

Für die IBM APL2-Workstation-Produkte ist seit Dezember 2000 zusätzlich zu den üblichen Fixes für gemeldete Fehler ein neuer Service Level verfügbar (CSD7 für das Windows-Produkt, CSD18 für OS/2, AIX/6000 und Sun Solaris). Für APL2/390 V2.2 sind entsprechende Ergänzungen via PTF ebenfalls erhältlich . Im folgenden Beitrag werden die wichtigsten Neuerungen vorgestellt, zu denen beispielsweise verbesserte Internet-Werkzeuge, eine optimierte ODBC-Unterstützung und neue Windows Standardsteuerelemente zählen. Neben einer Übersicht zu wichtigen Funktionsmerkmalen von IBM APL2 für Windows werden die Eigenschaften der Prozessoren und mitgelieferten Hilfsprozessoren für die Schnittstellen zu den System-Services aufgelistet und die Möglichkeiten der APL2 Runtime Environment for Windows skizziert.

Internet-Werkzeuge für alle APL2-Produkte

Im Arbeitsbereich NETTOOLS wurden zwei Verbesserungen eingebaut:

- Eine neue Konfigurationsvariable für den Index-Ursprung steuert diesen, wenn APL-Ausdrücke vom Client-PC aus ausgeführt werden (Einzelheiten im Kommentar zu CONFIGURE _HTTP_SERVER).
- Beim Server-Start werden mehr Konfigurationsinformationen angezeigt.

Benutzer mit APL2/390 V2.2 können NETTOOLS mit dem PTF für APAR Nr. PQ44399 auf den neuesten Stand bringen. Konsultieren Sie Ihren Systemprogrammierer bzw. den IBM TA. Hier noch ein weiterer Hinweis zu APL2/390: Mit dem Arbeitsbereich NETTOOLS wurde seinerzeit auch eine Windows DLL ausgeliefert, die es ermöglicht, von Nicht-APL-Anwendungen (z.B. EXCEL) oder auch von PC-APLs anderer Hersteller auf APL2/390 zuzugreifen.

REXX Prozessor 10 für APL2 auf Unix

Für APL2/6000 und APL2/Solaris ist (wie schon vorher bei APL2/Windows und APL2/OS2) jetzt der IBM REXX Prozessor 10 verfügbar. Er erlaubt nicht nur Funktionen und Unterroutinen von IBM Object REXX von APL2 aus aufzurufen, sondern er enthält verschiedene Datei-Dienstprogramme, die auch dann laufen, wenn kein IBM Object REXX installiert ist.

Neu für APL2 für Windows

ODBC-Unterstützung:

Das Windows-APL2 hat jetzt einen neuen Hilfsprozessor AP227, der eine Schnittstelle bildet zu allen Datenbanken und Programmen, die ihrerseits eine ODBC-Schnittstelle haben. Der AP227 hat den selben Befehlssatz und gleiche Syntax wie der schon lange vorhandene AP127 für DB2. Die gleiche SQL-Sprache wird benutzt, um mit Datenbanken zu kommunizieren. Entsprechende Funktionen im mitgelieferten Arbeitsbereich 2 SQL können mit dem AP227 benutzt werden, wenn die neue Variable SQL_AP auf „227“ gesetzt wird. Jedoch unterstützen nicht alle Anwendungen mit einer ODBC-Schnittstelle in vollem Umfang SQL.

Partial Line Marking für Sitzungsmanager und Objekt-Editor

Im Sitzungsmanager und im Objekt-Editor können Sie die bekannten Windows-Aktionen Copy and Paste für beliebige Textbereiche anwenden. Als Systemoption können Sie diese Funktionalität ein- oder ausschalten.

Neue Funktionen für AP145 (APL2 für Windows)

PICTURE-Eigenschaft

Die AP145 PICTURE-Eigenschaft erlaubt die Anzeige von Bildern auf Buttons, Checkboxen, Rechtecken und Frames. Gesetzt wird Sie durch Angabe des Bildnamens (Bitmap oder Icon) oder per Referenz auf eines der integrierten Bilder.

Cursor- und Cursor Position-Eigenschaft

Die AP145 CURSOR-Eigenschaft erlaubt die Anzeige eines Bildes, sobald sich der Cursor über einem Steuer-

Herunterladen der CSD

Die neuen CSDs kann man von den nachfolgenden Internetadressen herunterladen:

- APL2 für Windows: *www.apl-online.de oder www-4.ibm.com/software/ad/apl*
- APL2/RTE für Windows: *www.apl-online.de oder www-4.ibm.com/software/ad/apl*
- APL2/OS2: *www-4.ibm.com/software/ad/apl*

CSD-Beschaffung für APL2/6000 und APL2/Solaris:

Benutzer von diesen APL2-Systemen werden gebeten, Herrn *Klaus Drebenstedt* anzuschreiben (IBM Deutschland Informationssysteme, eMail: KWD@de.ibm.com). Sie erhalten dann auf Nachweis Ihrer IBM APL2-Lizenz eine CD mit den entsprechenden CSDs.

element befindet. Gesetzt wird die Eigenschaft analog zur PICTURE-Eigenschaft durch Angabe einer Cursordatei oder einer Referenz auf eine der 12 integrierten Cursordarstellungen. Die CURSOR POSITION-Eigenschaft ist ein zweielementiger Vektor, der die x- und y-Koordinate der Cursorposition in Pixeln wiedergibt.

Windows Standardsteuerelemente

Folgende Standardsteuerelemente können Sie ab dem kommenden CSD in Ihre Applikation einfügen:

- *Date, Time und Month Controls* - Schnittstelle für den Benutzer zur Auswahl eines Datums (Date) oder einer Zeitangabe (Time) bei gewünschter Formatierung bzw. die Möglichkeit, einen Zeitraum festzulegen (Month). Hierbei werden zahlreiche Eigenschaften unterstützt.
- *Listview Controls* - zur Darstellung von Listen, wobei jedes Element mit einem Bild oder einer Beschreibung versehen werden kann. Die Einträge können als Icon, Small Icon, List oder im Detail dargestellt werden. (vergleichbar mit den Darstellungsoptionen im Windows-Explorer) Ereignisse werden bei Auswahl von Einträgen ausgelöst.
- *Tab Controls* - Diese gleichen den Einteilungen eines Karteikartensystems oder Registern eines Kalenders. Diese Controls findet man eignen sich gut zur Definition von Benutzeroptionen oder Systemeinstellungen (vgl. MS Word oder MS Excel)

Dialogstil „SPLIT"

In diesem Dialogstil wird der Dialog in zwei Fenster gespalten, und zwar mit einem beweglichen „Teiler".

Statusleisten

Für AP145-Dialoge werden jetzt Statusleisten unterstützt. Diese werden durch die neuen Eigenschaften „STATUS VISIBLE", „STATUS DATA" und „STATUS PARTS" gesteuert.

Neue Demofunktionen im Arbeitsbereich 2 DEMO145

Die neuen AP145-Funktionen können durch neue abwandelbare Demofunktionen aus dem Arbeitsbereich DEMO145 in der öffentlichen Bibliothek 2 demonstriert werden:

- DEMO_DATETIME
- DEMO_LISTVIEW
- DEMO_MONTH
- DEMO_PICTURE
- DEMO_SPLIT
- DEMO_STATUS
- DEMO_TAB

Die Demofunktionen sollen, wie in APL2 üblich, als Beispiele dienen und sind abwandelbar für eigene Anwendungen.

APL2 für Windows Runtime Environment (RTE)

Wenn Anwendungen mit einem APL2 für Windows, das auf CSD7 aufgerüstet ist, für RTE paketiert werden, dann laufen diese Anwendungen auf RTE nur, wenn das RTE ebenfalls auf den neuen Service-Level aufgerüstet ist, auch wenn die vorstehend beschriebenen Neuerungen in der paketierten Anwendung nicht benutzt werden.

APL2 für Linux

Das IBM Labor Santa Teresa (STL) arbeitet an einem APL2 für Linux. Die Portierung des APL2 für AIX + Solaris nach Linux ist abgeschlossen. Der Grafik-AP, die Signalbehandlung und die Dokumentation sind noch in Arbeit.

STL hatte eine Umfrage verteilt, worin nach Hardware, Linux-Distributor, Angleichung an APL2 für Windows oder für Unix und nach den Preisen gefragt wurde, die potentielle Kunden wünschen bzw. bereit wären zu bezahlen. Es ist noch nicht bekannt, wie die Beteiligung war und wie die Auswertung aussieht.

Es wird erwartet, daß APL2 für Linux noch in diesem Jahr verfügbar wird.

Gesamtübersicht über APL2 für Windows

Nach den vielfachen Produktergänzungen, die im Laufe der vergangenen zwei Jahre durchgeführt wurden, ist es für die Anwender manchmal schwierig, den Überblick zu behalten. Als Service für den Leser folgt deshalb anschliessend eine Gesamtübersicht in drei Teilen:

- Übersicht APL2 für Windows
- Schnittstellen zu System-Services
- APL2 Runtime Environment for Windows

Übersicht zu IBM APL2 für Windows 1.07

- *Kompatibilität:* APL2 für Windows Version 1.07 ist mit folgenden IBM Produkten kompatibel:

⇨APL2 V2.2 für IBM/390-Großrechner
⇨APL2 für OS/2 V1.0 (CSD Level 18)
⇨APL2/6000 V1.2 (CSD Level 18)
⇨APL2 für SUN/Solaris V1.1 (CSD Level 18)

- *Grafische Benutzerführung:* Der Prozessor AP145 für grafische Benutzerführung in Anwendungen, der APL2-Sitzungsmanager und die integrierte APL2-Editoren nutzen die grafischen System-Services von Windows 95/98/ME® und Windows/NT/2000®.

- *Programmierschnittstellen für grafische Benutzerführung in Anwendungsprogrammen:* APL2 liefert mit dem AP145 einen grafischen Dialog-Editor und Zugriff auf „Windows"-Daten als Objekte im APL2-Arbeitsbereich. Umfassende Hilfen und Beispiele zur Entwicklung grafisch geführter Anwendungen und werden mitgeliefert. Windows-Standardsteuerelemente (Date, Time, Listview, Tab Controls) sowie Bilder können in APL2-Anwendungen und in den zugehörigen Steuerelementen benutzt werden.

- *High-Level-DDE-Unterstützung:* APL2 kann schnell und effektiv Daten mit anderen Anwendungen (zum Beispiel Tabellenkalkulationsprogrammen, Textverarbeitungssystemen) über DDE gemeinsam nutzen. Muster befinden sich im Arbeitsbereich DDESHARE, der mitgeliefert wird.

- *Mehrfache gleichzeitige APL2-Sitzungen:* Jede APL2-Sitzung arbeitet asynchron in ihrem eigenen Prozess und kann mit anderen APL2-Sitzungen auf der gleichen oder entfernten Maschinen (verbunden durch ein TCP/IP-Netz) Daten gemeinsam nutzen.

- *Druckerunterstützung:* Grafiken und Text, auch mit APL-Zeichen darin,

können auf jedem von Windows unterstützten Drucker gedruckt werden.

- *Grafik-Unterstützung:* Der universelle APL2-Grafik-Prozessor AP207 unterstützt neben dem eigenen auch verschiedene andere bekannte Grafikformate, z.B. .GIF und .JPG, und ist mit Werkzeugen kombinert, die voll kompatibel auch auf den anderen APL2-Workstation-Plattformen verfügbar sind. Der mitgelieferte Arbeitsbereich GRAPHPAK ist kompatibel für alle APL2-Plattformen.

- *Aufrufe von Programmen in anderen Sprachen:* APL2-Programme können über die Namensassoziierung jedes in einer DLL unter 32-bit System Linkage Conventions gespeicherte Programm aufrufen. Zu IBM REXX wird eine besonders effektive Verbindung hergestellt.

- *Große Arbeitsbereiche und Dateien als APL2-Variable:* APL2 nutzt die 32-bit-Architektur von Windows, die Anwendungen ermöglicht, bis zu 2 GB virtuellen Speicher zu adressieren. Arbeitsbereiche bis nahe an 2 GB werden unterstützt, jedoch ist die maximal mögliche Größe begrenzt durch den auf der benutzten Maschine vorhandenen realen Haupt- und Swap-Speicher. Darüber hinaus können ausserhalb des Arbeitsbereichs befindliche Dateien im Ganzen als APL2-Variable im Arbeitsbereich verarbeitet werden.

- *Werkzeuge und Arbeitsbereiche zur Anwendungsentwicklung:* Mit Hilfe des APL2-Objekteditors kann man nicht nur APL2-Funktionen komfortable editieren, sondern zusätzlich auch weitere APL2-Objekttypen (einschliesslich Alpha-Vektoren mit Zeilenendemarkierung sowie Vektoren von Vektoren).
  Arbeitsbereiche mit Beispiel- und Dienstprogrammen werden mitgeliefert, die direkt oder nach individuellen Bedürfnissen modifiziert verwendet werden können.
  Dazu gehört auch ein Arbeitsbereich, mit Hilfe dessen zwei Arbeitsbereiche miteinander verglichen werden können, sowie zahlreiche sehr nützliche APL2-externe Funktionen.
- *Online-Dokumentation (in Englisch):*
  Der „APL2 User's Guide", eine „APL2 Language Summary", „APL2 Programming: Using SQL" und der „GRAPHPAK User's Guide" werden online mitgeliefert. Für viele Einrichtungen ist zudem Online-Hilfe verfügbar.

- *Zugriff auf relationale Datenbanken:* Der AP127 liefert einen Schnittstelle zu IBM DB2 ® unter Verwendung der SQL-Sprache. Der AP227 bringt eine ODBC-Schnittstelle, die analog AP127 funktioniert und den Zugriff auf alle DB-Systeme und Anwendungen erlaubt, die ihrerseits eine ODBC-Schnittstelle besitzen. Mitgelieferte Beispielprogramme erleichtern die Anwendung.

- *Werkzeuge zum Schreiben von Hilfsprozessoren:* Muster für Hilfsprozessoren, „Include"-Dateien und eine Server-Schnittstelle erlauben eine einfache Implementierung von den eigenen Bedürfnissen angepaßten Hilfsprozessoren (einschließlich Client-Server-Prozessoren).

- *Performance-Überwachung:* Der TIME-Arbeitsbereich dient zur Analyse von Performance-Engpässen in Anwendungen.

- *Unterstützung von Landessprachen:* APL2 akzeptiert Befehle und gibt Nachrichten in verschiedenen Landessprachen aus. Die Tastatursteuerung paßt automatisch die APL-Tastaturbelegung den vom Benutzer dafür gewählten lokalen Standard-Einstellungen an. Sprachen mit Mehrfach-Byte-Zeichen, wie z.B. Japanisch, werden bei Grafiken, beim Drucken und bei den Schnittstellen zur graphischen Benutzerführung unterstützt.

- *Verteilte Verarbeitung, Client/Server und Internet:* APL2 bietet die Möglichkeit, Daten und Verarbeitung auf mehrere, über TCP/IP verbundene APL2-Systemen zu verteilen. Die APL2-Systeme unter TSO, CMS, OS/2, Windows, AIX/6000 und Sun Solaris haben alle die folgenden Einrichtungen:

  ⇨Gemeinsame systemübergreifende Variable zum Datenaustausch und zur Synchronisation
  ⇨TCP/IP „socket interface" zur Verbindung mit Nicht-APL-Anwendungen
  ⇨Programmsteuerung einer APL2-Sitzung durch andere Anwendungen oder Benutzer
  ⇨Sitzungsmanager für entfernte APL2-Systeme

  Mit diesen EInrichtungen, mit dem neuen Arbeitsbereich NETTOOLS und mit neuen Funktionen zur Zeichenübersetzung (u.a. SCAR = herstellerunabhängiges APL-Format, Unicode und DBCS) können auch TCP/IP-Client/Server- und Internet-Verbindungen zu APL-Anwendungen anderer Hersteller sowie zu Nicht-APL-Anwendungen hergestellt werden. In letztgenannten erlaubt eine neue unabhängige APL2-Tastatur-Unterstützung (standalone keyboard handler) die Benutzung der APL2-Tastatur.

- *Jahr-2000-Fähigkeit:* APL2 für Windows ist in vollem Umfang Jahr-2000-fähig.

- *APL2-Schnittstellen zu System-Services:* APL2-Schnittstellen zu System-Services werden durch synchron arbeitende assoziierte Prozessoren mit Namensassoziierung und durch asynchrone Hilfsprozessoren mit gemeinsamen Variablen bereitgestellt.

APL2-Schnittstellen zu System-Services

APL2-Schnittstellen zu System-Services werden durch synchron arbeitende assoziierte Prozessoren mit Namensassoziierung und durch asynchrone Hilfsprozessoren mit gemeinsamen Variablen bereitgestellt. Mit ihnen entwickelte Anwendungen sind weitgehend portabel über alle APL2-Plattformen.

Als assoziierte Prozessoren werden geliefert:

P10 zum Aufruf von REXX-Routinen und eingebauten Funktionen, die außerhalb des APL2-Arbeitsbereiches laufen, aber wie APL2-Funktionen behandelt werden können. Verschiedene P10-Routinen werden mitgeliefert, u.a. zum Lesen und Schreiben von und in Dateien (z.B. DeltaFV und DeltaFW analog den APL2-Funktionen FV und

FW im Arbeitsbereich FILE). Die mitgelieferten Funktionen laufen, ohne daß IBM Object REXX installiert sein muß. APL2 Arrays können mit P10 auch als REXX-Programme ausgeführt werden.

P11 zum Aufruf von DLL-Routinen, die außerhalb des APL2-Arbeitsbereiches laufen. Es können unabhängig von der Programmiersprache alle Programme angesprochen werden, die in einer Windows-DLL gespeichert sind und den „32bit Linkage Conventions" entsprechen. Verschiedene P11-Routinen werden mitgeliefert, unter anderem zum Lesen und Schreiben von ganzen Dateien (einschließlich Steuerzeichen) als APL2-Alphavektor, zum Umwandeln von Byte-Sätzen in APL2-Strukturen und umgekehrt (ATR, RTA, PFA) und GETENV zum Abfragen von Werten von Variablen der Systemumgebung. Dokumentation und Musterdateien zur Entwicklung eigener externer, von P11 aufzurufender Routinen werden mitgeliefert.

P12 zur Verarbeitung von Dateien als APL2-Variable: Hierbei sind APL- und Text-Dateien möglich. Sie befinden sich außerhalb des APL2-Arbeitsbereiches und können größer sein als dieser. Sie können mit den zahlreich mitgelieferten APL2-Elementar-Funktionen verarbeitet werden, als ob sie sich im Arbeitsbereich befinden würden. Dokumentation und Muster zur Entwicklung eigener P12-Funktionen werden mitgeliefert. Bezüglich der Portabilität von Anwendungen mit P12 vom/zum Großrechner bestehen kleine Einschränkungen.

Mitgelieferte Hilfsprozessoren:

AP100 für Host-System-Befehle: übermittelt Host-System-Befehle aus APL2-Anwendungen.

AP101 für Alternativ-Eingabe (Stack): zur Erstellung von programmierten Alternativ-Eingaben an APL2.

AP119 TCP/IP Socket API: leitet Aufrufe an TCP/IP weiter, das zur Kommunikation über Netze dient.

AP124 zur Textanzeige: ermöglicht Text-Fenster unter Kontrolle der Anwendung. Arbeitsbereiche für Cover-Funktionen und Demos sind beigefügt.

AP127 SQL Prozessor: zum Zugriff auf IBM DB2 (auch auf verteilte DB2-Systeme) über die SQL-Sprache.

AP145 für grafische Benutzerführung in Anwendungen: ermöglicht den Zugriff auf Services der grafischen Benutzerführung sowie auf Standardsteuerelemente (Date, Time, Listview, Tab Controls) von Windows. Bilder können in APL2-Anwendungen und in den zugehörigen Steuerelementen benutzt werden. Werkzeug- und Demo-Arbeitsbereiche sind beigefügt.

AP207 Universeller Grafikprozessor: zur Erzeugung von Grafiken, Texten und Bildern hoher Qualität über eine einfache Befehlssyntax (mit beigefügten Beispiel-, Demo- und GRAPHPAK - Arbeitsbereichen); unterstützt gängige Grafikformate, unter anderem GIF und JPG.

AP210 Dateiprozessor: zum Schreiben und Lesen von Dateien des Betriebssystems. Diese können Texte, binäre Daten und APL2-formatierte Objekte enthalten.

AP211 APL2 Objekt-Bibliotheks-Prozessor: zum Schreiben und Lesen von APL2-Strukturen über ihre Namen. Die damit erzeugten Dateien sind portabel auf alle anderen APL2-Plattformen (und umgekehrt).

AP227 ODBC-SQL-Prozessor: zum Zugriff auf alle DB-Systeme und Anwendungen, die ihrerseits eine ODBC-Schnittstelle besitzen. Mitgelieferte Beispielprogramme erleichtern die Anwendung.

AP488 APL2-GPIB-Prozessor (IEEE 488 General Purpose Interface Bus): zum Datenaustausch mit Geräten, die an diesen Bus angeschlossen werden können, z.B. Plotter, Messgeräte usw. Es werden die Adapterkarten und Treiber von National Instruments unterstützt. Der AP488 ist kompatibel mit dem gleichnamigen Prozessor von APL2/PC(DOS).

IBM APL2 Runtime Environment for Windows

Dieses Zusatzprodukt zum APL2 für Windows bietet eine preiswerte Alternative, um APL2-Anwendungen ohne Entwicklungsumgebung nur laufen zu lassen und ihren APL2-Code gleichzeitig durch Einkapseln vor Manipulation und unbefugtem Zugriff zu schützen.

Das APL2 Runtime Environment besteht aus zwei Teilen, dem sogenannten *Workspace Packager* (Paketierer für Arbeitsbereiche) und der *Runtime Library* (Ausführungsbibliothek). Die Hardware-Voraussetzungen sind mit der APL2 für Windows identisch.

Mit dem Workspace Packager kann der Anwendungsentwickler aus einem APL2-Arbeitsbereich eine Windows DLL (Dynamic Link Library = paketierter Arbeitsbereich) erzeugen. Für den Betrieb des Workspace Packager ist das volle APL2 für Windows (V.1.07 oder höher) und Windows NT 4.0 bzw. Windows 2000 Voraussetzung. Er arbeitet *nicht unter Windows 95/98*.

Die Runtime Library wird auf der Maschine des Anwenders installiert und umfaßt diejenige Untermenge von APL2 für Windows, die erforderlich ist, um DLLs laufen zu lassen, die mit dem Packager erzeugt wurden. Die Runtime Library läuft unter Windows 95/98 oder NT. Sie enthält den APL2-Interpreter, die assoziierten Prozessoren und Hilfsprozessoren sowie die von IBM mitgelieferten externen Funktionen. Nicht enthalten sind die Komponenten der Entwicklungsumgebung, wie Sitzungsmanager, Editoren, öffentliche Arbeitsbereiche und Dokumentationen.

Paketierte Anwendungen, die mit der Runtime Library laufen sollen, müssen mit einer „Latent Expression" starten und unterliegen den folgenden Beschränkungen:

- Alle Systembefehle sind abgeschaltet.
- Zeilen-Ein/Ausgabe (Quad und Quote-Quad) ist nicht unterstützt. (Jede Anforderung für Zeileneingabe bei leerem Stack verursacht eine Beendigung von APL2.)

Paketierte Arbeitsbereiche laufen sowohl unter der Runtime Library als auch unter dem vollen APL2 für

Windows. Diese Arbeitsbereiche können wahlweise in einem Paket zusammen mit der Runtime Library verteilt werden oder die Runtime Library wird separat beschafft und installiert und nur der paketierte Arbeitsbereich wird verteilt.

Wenn Anwendungen mit einem APL2 für Windows, das auf CSD7 aufgerüstet ist, für RTE paketiert werden, laufen diese auf RTE nur, wenn dieses ebenfalls auf den neuen Service-Level aufgerüstet ist, auch wenn die Neuerungen von CSD7 in der paketierten Anwendung gar nicht benutzt werden.

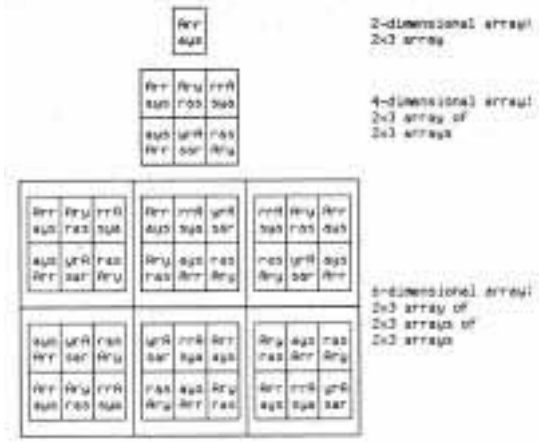
Das Produktpaket APL2 Runtime Environment für Windows enthält zwei CD-ROM (eine mit dem Workspace Packager und der Online-Dokumentation „APL2 Workspace Packager User's Guide", und eine mit der Runtime Library). Desweiteren sind ergänzende Installations-Lizenzen erhältlich.

Die IBM Deutschland GmbH hat Dittrich & Partner Consulting GmbH (DPC) mit der Abwicklung der Bestellungen und Lieferungen beauftragt. Die IBM bietet (unter gewissen Bedingungen für die Fehlermeldung) zentralen Service über Post, Fax, e-mail oder ftp-Server. DPC bietet darüber hinaus auf Anfrage Hotline Service zur Anwendungsberatung.

Kontakt:

Axel Güth (eMail: AGueth@csi.com), Dittrich & Partner Consulting GmbH (DPC), Augsburg, Solingen, Potsdam. Internet: www.dpc.de

Der vorliegende Text entstand aus Bearbeitungen von englischen Vorlagen von David Liebtag und Nancy Wheeler, beide STL, und deutschen Vorlagen von Dr. Christiane Sonnenschein, DPC Solingen.



APL2001 : An Arrays Odyssey
International conference on array programming languages to be held on June 25–28, 2001 at Yale University in New Haven, Connecticut, U.S.A.

Wolfgang Strasser

PAngV oder Newton Revisited

Zur Berechnung des Effektivzinses

Mit dem Newton-Verfahren kann man für eine gleichbleibende Zahlungsperiode den effektiven Jahreszins ausrechnen, den auch die Hausbank angibt. Wählt man aber andere (unterjährige) Zeiträume, so stimmen die Ergebnisse nicht mehr überein. Dies liegt daran, dass die deutschen Banken bislang nach Vorschriften rechnen, die durch die Preisangabeverordnung (PAngV) gesetzlich vorgegeben sind. Diese Verordnung soll nach dem Willen der EU durch Regeln der ISMA (International Securities Market Association) ersetzt werden und die Definition des Effektivzinses in der EU vereinheitlichen. Die Effektivzinsdefinition nach ISMA ist seit dem 1. Januar 2001 auch in Deutschland verbindlich. Man findet allerdings auf breiter Front immer noch Angaben nach PAngV, da der Effektivzins nach dieser Methode höher ist. In diesem Beitrag werden Vorgehensweisen aufgezeigt, wie der Newton Algorithmus auf mehr oder minder unregelmässigen Zahlungsperioden angewandt werden kann.

Ausgangspunkt

Um die Güte von Investitionen zu messen, greift man üblicherweise auf den Begriff des *effektiven Jahreszinses* zurück, auch innerer Zinssatz oder Internal Rate of Return (IRR) genannt. Dabei stellt man sich vor, dass die Anfangsinvestition K_0 gerade gleich dem Barwert der zukünftigen Rückflüsse R_i ist, d.h. gleich der Summe der zukünftigen Zahlungen R_i, die mit dem Effektivzinsfuss i diskontiert wurden. Dies ergibt die allgemeine Gleichung:

$$0 = -K_0 + R_1 d + R_2 d^2 + \ldots + R_n d^n$$

$$d = \frac{1}{1 + i/100} \quad ; \quad i - \textit{Effektivzins in \%}$$

Dabei kann K_0 der Kurs einer Aktie oder eines festverzinslichen Wertpapiers sein, das eine Privatperson kauft, oder - wenn der Investor eine Bank ist - könnte K_o auch die ausgezahlte Summe eines Hypothekendarlehens sein. Die später erfolgenden Rückzahlungen R_i entsprechen dann Zahlungen für Dividenden oder Zinsen, respektive Zahlungen für Hypothekentilgung und -zins. Die R_i sind in vielen praktischen Fällen immer positiv, es können aber auch negative Werte auftreten; man denke nur an die Beteiligung an einem Unternehmen oder an einem Aktienportefeuille.

Das Newton-Verfahren

Die angeführte Gleichung beschreibt die Nullstellen eines Polynoms f(d) n-ten Grades: es gilt also diese zu bestimmen. Hierfür verwendet man in sehr effizienter Weise das Newton Verfahren. Sein Grundgedanke, der nicht nur für Polynome gilt, besteht darin, die Tangente an eine differenzierbare Funktion $f(x_1)$ für einen beliebigen Startwert x_1 so zu verlängern, bis sie die x-Achse im Punkt x_2 schneidet. Als nächster Schritt wird die Tangente an $f(x_2)$ betrachtet, dann an $f(x_3)$ usw. Das Verfahren konvergiert schnell; es scheitert lediglich, wenn $f(x_k)$ zufällig ein Extremum der Funktion f(x) ist, da dann die Tangente parallel zur Abszisse verläuft und diese nicht mehr schneidet.

Formal ergibt sich an der Nullstelle x_j+d die Taylor Entwicklung:

$$0 = f(x_j + \delta) = f(x_j) + f'(x_j)\delta + \frac{f''(x_j)}{2}\delta^2 + \ldots$$

und somit für kleine d im linearen Term:

$$\delta = -\frac{f(x_j)}{f'(x_j)}$$

Da es sich bei f(x) um ein Polynom handelt, lässt sich sowohl f(x) als auch die Ableitung f'(x) in APL einfach darstellen (Zeilen 11 und 12), und das Programm zur Berechnung des inneren Zinsfusses nimmt folgende Gestalt an:

```
     ∇ R←(LA)IRRc RA;RF;IP;FP           ⍝ Internal Rate of Return
[1]                                     ⍝ stetiger Zins
[2]   ⍝ Gerber (?)  Iteration
[3]   ⍝ LA - Zeitvektor (regel- oder unregelmässige Abstände) ⍴LA = ⍴RA
[4]   ⍝ RA - Ergebnisvektor (= Polynomkoeffizienten): ¯vv a1 a2 ..
[5]
[6]   ⍝------------------------------------
[7]
[8]   R←0,03                            ⍝ Startwert 3%
[9]
[10]  ⍎(0=⎕NC'LA')/'LA←1'               ⍝ Default LA = 1 Jahresabstand
[11]  ⍎(0=⍴LA)/'LA←LA×¯1+⍳⍴RA'          ⍝ LA Skalar =
                                              aequidistante Zahlungen
[12]
[13]  IP←-↑RA ◇ FP←+/1↓RA          Investment, Summe zukünftiger Zahlungen
[14]  RF←{⍵+(⍵-IP÷FP)÷⍵+/((1↓RA)÷FP)×*-(1↓LA)×⍵}
[15]                                          ⍝ Rekursionsformel
[16]
[17]  R←(RF u.rptuntil 1)R
[18]
[19]  →(1>R)/OO
[20] ERR:→⍴0,⍴⎕←'keine Amortisation, neg. Zins: ',5 u.fvh R
[21] OO:R←3 u.rd 100×¯1+*R
     ∇
```

Der Operator «rptuntil» (wiederhole bis zur Abbruchgrenze), sowie die Hilfsfunktionen «rd» (Runden) und «fvh» (Format) befinden sich im Namespace u und werden im Anhang dieses Artikels dargestellt.

Da wir von nur einem einzigen Effektivzins i ausgehen, betrachten wir implizit äquidistante Zeitpunkte t, an dem die Zahlungen R_t vorgenommen werden. Beträgt die Distanz zwischen den Zahlungen genau ein Jahr, so können wir mit dem Newton-Verfahren den effektiven Jahreszins ausrechnen, den uns auch die Hausbank angibt. Wählen wir aber andere (unterjährige) Zeiträume, so stimmen die Ergebnisse nicht mehr überein.

PAngV und ISMA

Dies liegt daran, dass die deutschen Banken – bislang – nach Vorschriften rechnen, die durch die Preisangabeverordnung 1985, »PAngV«, gesetzlich vorgegeben sind. Diese Verordnung soll nach dem Willen der EU durch Regeln der ISMA (International Securities Market Association) ersetzt werden und die Definition des Effektivzinses in der EU vereinheitlichen. Je nach Quelle ist die Effektivzinsdefinition nach ISMA bereits seit Mitte 2000 bzw. seit dem 1. Januar 2001 auch in Deutschland verbindlich. Man findet allerdings auf breiter Front immer noch Angaben nach PAngV: schliesslich ist der Effektivzins nach dieser Methode auch höher.

Die ISMA-Regeln fordern ganz einfach eine taggenaue Verzinsung mit Zinseszins. Dies entspricht dem intuitiven Verständnis von unterjährigem Zins, wenn man – für eine vorgegebene jährliche Zinsrate r = 1+i – den Zinsertrag eines Tages als $\sqrt[365]{r}$ definiert.

Beispiel:

Jemand tätigt am 1.1.2000 eine Investition von 1'000 €, er erhält am 15.11.2000 (also am 319ten Tag des Jahres) eine Rückzahlung von 400 € und am 31.12.2001 eine weitere Rückzahlung in Höhe von 800 €. Dann ist die Gleichung zu lösen:

$$0 = 1'000 + 400 \times (1+i)^{-319/365} + 800 \times (1+i)^{-730/365}$$

Dies führt zu einem jährlichen Effektivzins i von 12 %.

Im letzten Abschnitt dieses Artikels werden Vorgehensweisen aufgezeigt, wie der Newton Algorithmus auf solche mehr oder minder unregelmässigen Zahlungsperioden angewandt werden kann.

Die Regeln von PAngV sind zwar in einem Gesetzestext dokumentiert, aber dieser bleibt einem normalen Menschen völlig unverständlich. Ich bezweifele, dass er für Juristen (»iudex non calculat«) eher zugänglich ist. Ansonsten gibt sich das Internet eher zugeknöpft: einige Vorschläge zur Weltverbesserung, einige Sharewareprogramme, aber wenig zum eigentlichen Algorithmus. Soweit ich verstanden habe, läuft PAngV darauf hinaus, unterjährige Zahlungsströme, die nur am Anfang und am Ende eines Investitionsprozesses vorkommen, mit der einfachen Verzinsung zu bedenken. Zu diesem Zweck besteht jedes Finanzjahr aus 12 Monaten à 30 Tagen, also aus insgesamt 360 Tagen. Der Zahlungszeitpunkt wird nicht taggenau berechnet, sondern auf den Beginn desjenigen Monats gelegt, in dem eine Zahlung getätigt wird. Dann erfolgt eine lineare Verzinsung bis zum Jahresende (die dann ganzjährig diskontiert wird).

Beispiel:

Für die oben dargestellten Zahlungsströme ergibt sich folgende Gleichung:

$$0 = -1'000 + 400 \frac{(1 + i\,\frac{60}{365})}{1+i} + 800 \times (1+i)^{-2}$$

Dies führt zu einem effektiven Jahreszins von 12.15 %

Stetige Verzinsung

Der gängigen Finanzmarkttheorie liegt die Vorstellung zugrunde, dass die Verzinsung ein stetiger Prozess ist, und nicht in diskreten Sprüngen erfolgt, seien sie jetzt jährlich oder täglich. Die stetige Verzinsung ist durchaus praktisch handhabbar, und sie ist auch unbedingt dann anzuwenden, wenn «richtige» Finanztheorie ins Spiel kommt und man z.B. Optionspreise berechnen will.

Die lokale Stadtsparkasse würde sich allerdings nicht über stetige Verzinsung freuen: Entspricht doch eine jährliche Rendite von 15 %, die sie dem Käufer von Fondsanteilen verspricht, einer stetigen Verzinsung von mageren 13.98%. Dies folgt aus der Definition des Jahreszinses bei stetiger Verzinsung:

$$115 = \int_0^1 e^{0.13976\,t}\,dt$$

Korollar ergibt sich somit aus einem Jahreszins die stetige Verzinsung durch:

$$i_S = \ln\left(1 + i_{jahr}\right).$$

Der grosse Vorteil der stetigen Verzinsung liegt m.E. darin, dass alle Zeitperioden gleichberechtigt nebeneinander betrachtet werden können und auf äquidistante Zahlungszeitpunkte verzichtet werden kann. Gerber (Life Insurance Mathematics) gibt eine schnelle Rekursion an, die beliebige Zahlungszeitpunkte zulässt. Allerdings müssen bei diesem Algorithmus *alle* Rückzahlungen Ri positiv sein.

Dieser Algorithmus wird durch nachstehendes APL Programm IRRc beschrieben.

Newton FAPP/FMPP

Für die Praxis der Renditeberechnung bei taggenauen Zinsbestimmung kann man das Newton Verfahren, so wie es oben vorgestellt wurde, leicht anwenden, indem man als Standardzahlungszeitraum 1 Tag festsetzt.

Betrachtet man ein Portefeuille von Investitionen, in welchem für unterschiedliche Wertpapiere sehr häufig Zahlungen anfallen, so ist diese Vorgehensweise sicher auch angemessen. Betrachtet man aber lediglich ein Rentenpapier, das einmal im Jahr einen Coupon zahlt, so erhält man

als Vektor der Zahlungen gewissermassen einen «sparce vector», bei dem auf einen Wert (Couponauszahlung) 364 Nullen (keine Zahlung) kommen. Hier ist die Wahl der Periode 1 Tag vielleicht doch zu aufwendig. Schliesslich muss letztlich der Anwender diesen Zahlungsvektor erst generieren.

Vielleicht nicht ganz FAPP (for all practical purposes), aber doch FMPP (for most practical purposes) bietet sich daher folgende Überlegung an: Die meisten Transaktionen folgen ihrer Struktur nach einem regelmässigen Zahlungsmuster, lediglich das anfängliche Investment macht eine Ausnahme (und ggf. das Desinvestment zum Ende). Von daher würde es – FMPP – ausreichen, die erste Periode zwischen Investment K_0 und erster Rückzahlung R_1 angemessen in den Griff zu bekommen. Dazu kann man die eingangs angeführte Gleichung 1 wie folgt umformulieren:

$$0 = -K_0 + d^{k/365}(R_1 d^0 + R_2 d^1 + \ldots + R_n d^{n-1})$$

Die Variable k gibt dabei die Anzahl der Tage zwischen Investment K_0 und erster Rückzahlung R_1 an. Ist k = 365, also ein volles Jahr, so ist man zurück bei der Ausgangssituation von Gleichung 1. Die zeitlichen Differenzen zwischen allen Rückzahlungen R_j betragen jeweils ein volles Jahr.

Auf diese Situation lässt sich ebenfalls das Newton Verfahren anwenden und in APL wie folgt formulieren:

```
    ∇ B←(LA)IRRt RA;RAt;del;newton;f;f'        ⍝ Internal Rate of Return
                                                  UNTERJÄHRIG
[1]
[2]   ⍝ Newton-Raphson Iteration - Mit UNTERJÄHRIGEM Anfangsglied
[3]   ⍝ LA - Anzahl der Tage bis 1. Rückfluss;
              z.B. 46 Tage bis zum 31.12.
[4]   ⍝    - Startwert xx + X.XX (5.35%)
[5]   ⍝ RA - Ergebnisvektor (= Polynomkoeffizienten): "xy a1 a2 ...
[6]
[7]   ⍝-----------------------------------
[8]
[9]   ⍎(0=⎕NC'LA')/'LA←365 5'               ⍝ Default: 365 Tage=1 Jahr, Zins 5%
[10]  ⍎(0≠ρLA)/'LA←LA,5'                    ⍝ wenn nur delta, + Zins 5%
[11]  del←(1⊃LA)÷365 ⋄ LA←1+0.01×2⊃LA           ⍝ trennen
[12]
[13]  RAt←↑RA ⋄ RA←1↓RA                         ⍝ Aufspalten der Koeffizienten
[14]
[15]
[16]  f←{RAt+(⍵*⍺)×⍵⊥⌽RA}                       ⍝ 'Polynom' mit Koeff. ⌽RA
                                                    + Zinsfaktor ⍵*⍺
[17]  f'←{(⍵*⍺-1)×⍵⊥[illegible]⌽RA}              ⍝ Derivative 'Polynom'
[18]
[19]  newton←{⍵-(⍺ f ⍵)÷(⍺ f' ⍵)}               ⍝ Newton Algorithmus
[20]
[21]  B←del(newton ⍺.rptuntila)LA
[22]
[23]  →(1=B)/GO
[24]  ERR:→0,ρ⎕←'keine Amortisation, neg. Zins!',5 ⍺.fsh B
[25]  GO:B←3 ⍺.rd 100×(÷B)-1
```

Man beachte, dass in diesem Programm der Operator «rptuntila» (Wiederhole bis zur Abbruchgrenze) aufgrund des vorhandenen linken Arguments α anders definiert ist als das «rptuntil» im ersten Programm IRR.

Beispiel:

Das bislang gewählte Beispiel wird wie folgt abgeändert:

Jemand tätigt am 15.11.2000 eine Investition von 1'000 €, er erhält am 31.1.2001 (also am 77ten Tag nach seinem Investment) eine Rückzahlung von 400 € und ein weiteres Jahr später, am 31.1.2002 eine weitere Rückzahlung in Höhe von 800 €. Dann ist die Gleichung zu lösen:

$$0 = -1'000 + (1+i)^{-77/365} \times \left(400 + 800 \times (1+i)^{-1}\right)$$

Dies führt zu einem Effektivzins von i = 23.82 %

Anhang

(Vgl. Abschnitt „Das Newton-Verfahren“)

```
    ∇rptuntila←{                        ⍝ OPERATOR:  repeat until ⍺⍺ ⍵ < EPS
[1]
[2]        EPS←0.0000001                ⍝ Precision Limit
[3]        EPS≥|⍵-⍺ ⍺⍺ ⍵:⍵              ⍝ until ⍵⍵ Condition satisfied
[4]        ⍺ ∇ (⍺ ⍺⍺ ⍵)                 ⍝ Apply ⍺⍺ ⍺⍺ ... until
[5]    }
    ∇

    ∇rptuntil←{                         ⍝ OPERATOR:  repeat until ⍺⍺ ⍵ < EPS
[1]
[2]        EPS←0.0000001                ⍝ Precision Limit
[3]        EPS≥|⍵-⍺⍺ ⍵:⍵                ⍝ until ⍵⍵ Condition satisfied
[4]        ∇ ⍺⍺ ⍵                       ⍝ Apply ⍺⍺ ⍺⍺ ... until
[5]    }
    ∇

    ∇ R←(LA)rd RA;K
[1]
[2]   ⍝-----------------------------------------------
[3]   ⍝  Rundet gegebenen Ausdruck RA auf LA Stellen
[4]   ⍝-----------------------------------------------
[5]
[6]   ⍎(0=⎕NC'LA')/'LA←9'                      ⍝ max. Darstellung ohne E-10 etc.
[7]   ⍝                                           max. 9 Stellen!
[8]   R←(1÷K)×⌊0.5+RA×K←10*LA
    ∇
```

Kontakt:

Wolfgang Strasser kann unter folgender eMail-Adresse erreicht werden: strasser@gmx.net.

Manchmal importiert man aus Excel »numerische« Tabellen, in welchen einzelne Zellen anstatt der 0 ein Leerzeichen oder keinen Eintrag aufweisen.

```
1       1
    2           3
3   4       1
7   5   2
```

Die Struktur dieser Tabelle ist geschachtelt mit Tiefe 2 (oder ¯2) und enthält an den Stellen ohne Eintrag den (eingepackten) leeren Vektor.

```
      ⎕display ,data
```

[illegible]

Um mit der Matrix weiterrechnen zu können, müssen die leeren Vektoren zuerst durch 0 ersetzt werden. Dies gelingt – erstaunlicherweise - mit Position »⍳« und Zahlen modulo Max+1:

```
      (1+7)|(⍳7)⍳data
1 0 1 0 0
0 2 0 0 3
3 4 0 1 0
7 5 2 0 0
```

Die Lösung setzt allerdings voraus, dass man das grösste Element der Matrix, also hier 7, kennt, was in der Regel gerade nicht der Fall ist. Also ist zuerst der Prototyp des leeren Vektors zu bestimmen: es handelt sich dabei um den geschachtelten leeren Character-Vektor:

```
      data[1;2]≡⊂0⍴' '
1
```

Jetzt kann man auch die leeren Vektoren eliminieren und aus dem verbleibenden numerischen Vektor das Maximum bestimmen.

```
      ⌈/(,data)~⊂0⍴' '
7
```

Damit erhalten wir folgendes kleines Programm:

```
pos1←{                ⍝ Position ⍳ ersetzt leeren Vektor durch 0
           α←⌈/(,⍵)~⊂0⍴' '
    (1+α)|(⍳α)⍳⍵
}
```

Beginnt man gleich mit dem Prototypen des leeren Vektors, so kann man ein alternatives Programm mit Reduktion und Expansion zur Problemlösung heranziehen:

```
pos2←{                ⍝ Position ⍳ ersetzt leeren Vektor durch 0
    ⍺←(,⍵)∊⊂0⍴' '
     (⍴⍵)⍴⍺\(~⍺)/,⍵
}
```

Bei dem Programm pos2 ist wegen \ / allerdings zu beachten, dass in keiner Zelle eine Null stehen darf; also eher eine Bedingung für konsequente Datenlieferanten.

Anmerkung des Redakteurs:

Wenn der Datentyp numerisch ist, kann man mit der Funktion: +/¨ leere Felder in = umwandeln. Dies funktioniert in allen modernen APL-Produkten.

Kontakt:

Wolfgang Strasser ist erreichbar unter der eMail-Adresse: strasser@gmx.net

Martin Barghoorn

Fahrrad und Computer

Technik, die uns bewegt

Was hat das Thema Fahrrad in einer aufstrebenden Computerzeitschrift zu suchen? Diese Frage kann schnell beantwortet werden: Fahrräder sind ein schöner Ersatz für Computer oder andersherum, besonders im Winter kann man gut das Fahrrad durch den Computer ersetzen. Alles unter dem gemeinsamen Motto: Überwindung von Raum und Zeit bei geringster Anstrengung. Es handelt sich also zweifellos um zwei substitutive Güter, die auch eine additive Komponente besitzen: den Fahrradcomputer. Der ist aber in diesem Beitrag nicht gemeint.

Bekannte Probleme

Die Frage ist erlaubt, was anstrengender oder schöner ist: Fahrrad bei Gegenwind oder Computer bei Serverstillstand. Wenn zum Beispiel der Computer oder das Netz oder beide streiken, kann man sich z.B. auf dem Fahrrad den Frust abreagieren oder auch wichtige und unwichtige Nachrichten in magnetischer oder Papierformatierung über eher nur geringe Entfernungen persönlich transportieren. Das ist bestimmt gesund und stärkt die Kommunikation im Kiez. Genausogut kann ein unterwegs platter Reifen die Kontaktchancen sehr verbessern (Mitleideffekt).

Luststeuer

Fahrrad und Computer sind also zwei wichtige Hilfsmittel des Menschen, die zwar direkt nichts miteinander zu tun haben, die aber für eine große Menge unserer Zeitgenossen von herausragender Bedeutung sind. Beide Dinge bereiten eine Menge Spaß und haben Suchtpotenzial. Hinzu kommt, dass sie vergleichsweise wenig kosten und vom Finanzamt auch noch begünstigt werden.

Den Regierungen, ständig auf der Suche nach neuen Einnahmequellen, werden diese Aspekte auf die Dauer nicht verborgen bleiben. Hier stutzt der geneigte Leser: wollen uns die Grünen nach dem Auto auch noch den Computer vermiesen? Man(n) möchte doch keines davon mehr hergeben. Eher schon die Näh- und/oder die Kaffeemaschine. Gemach, gemach, noch ist es nicht so weit, zumindest was die Fahrradsteuer anbetrifft! Doch für den Computer liegen entsprechende Pläne bereits griffbereit in der Schublade des Finanzministers. Als APLer schlagen wir in diesem Falle als Alternative sogleich die Besteuerung der Programmcodes vor, jede Zeile kann extra kosten und Schleifen natürlich je nach Durchlauf mehrfach.

My Home is my Castle

Der Home-Trainer hat sogar den Home-Computer überlebt. Wer aktuell über ein reales Fahrradobjekt mit zwei temporär aufblasbaren Reifen verfügt, kann sich in Echtzeit im Freien abstrampeln. Dabei spürt man sogleich den Fahrtwind direkt im Gesicht und kann so den Widerstand des Computers gegen jegliche Art von Programm, also Bitbewegung, selbsterfahrend nachempfinden. Deshalb empfiehlt es sich immer, vor dem Panorama der geschmackvollen Landschaftstapete auf dem Heim-Fahrrad virtuelle Fahrten zu unternehmen. Hierbei fällt erfreulicherweise die fremd verschuldete Umweltverschmutzung fast völlig weg.

Eine reale Windgefahr besteht erfahrungsgemäß am schlimmsten in unseren nördlichen Landesteilen, speziell auf den gefürchteten Deichkronen-Touren. Hier hat schon manche(r) RadfahrerIn einen Rückwärtsgang am Fahrrad schmerzlich vermisst.

Balance und Absturz

Fahrräder haben zwei Räder, deren ständige Drehung den Umfall verhindert.

Computer haben dagegen meist nur ein Rad (in der Festplatte), deshalb ist es hier noch schwieriger, die Balance zu halten und den Absturz zu verhindern. Weitere Festplatten können die Absturzgefahr kaum vermindern, denn die Gefahr geht vom jeweiligen Betriebssystem selbst aus. Die Potenz des Absturzes ist bei Computern sehr groß, sie besteht stets und ständig. Der Permanentspeicher heißt eigens so, weil er die Aufmerksamkeit auf diese permanente Gefahr lenken will.

Wenn wir es mit dem Fahrrad eilig haben oder auch aus Übermut, legen wir uns in die Kurve und fallen dennoch nicht um. Dagegen widersetzt sich der Computer den Verlockungen des Geschwindigkeitsrausches. Die Festplatte macht ihrem Namen alle Ehre und hält alle Daten fest, dass die Bilder ganz festfrieren und wir schon bald nach einem neuen, schnelleren Prozessor Ausschau halten müssen.

Technik mit Herz

Der Prozessor des Fahrrads ist das Tretlager mit den Kurbeln und Pedalen. Hier wird das relativ regelmäßige zweidimensionale Auf und Ab des Menschen in Form der körperlichen Muskelkraft in Rotationsbewegung umgesetzt und so die dritte Dimension erschlossen. Aus der simplen zweidimensionale Spalten-Tabelle, deren Inhalte zudem noch stark voneinander abhängig sind, werden mit dem Fahrrad auf wunderbare Weise die Hyperplanes des 3-D Arrays generiert und lustvoll fliegt die Landschaft an uns vorbei. Man träumt und hat Erinnerungen. Ortsveränderung schult das räumliche Gedächtnis, das man für die Arrayprogrammierung dringend benötigt.

Damit auch mathematisch vorbelastete Leser verstehen, was gemeint ist, lässt sich der Sachverhalt funktional wie folgt darstellen:

```
0 = f(↑↓ | ↑↓)

f = SIN (POWER)

POWER > WIND + HANG + REIBUNG
```

Mit diesem velozipedisch-mathematischen Grundwissen können wir in weite Ebenen vorstossen und/oder steile Denkgipfel erklimmen. Hierfür braucht es natürlich noch den angemessen getackteten Bus: die gut geölte Kette. Was sagte doch schon diesbezüglich die Mutter mit verzweifeltem Unterton in der Stimme: "Wagenschmiere, die krieg ich nicht raus, schon wieder eine neue Hose fällig!"

An der Peripherie

Während das Hinterrad mittels der Kette über das Ritzel angetrieben abrollt und somit die eigentliche Bewegung des Gefährtes bewirkt wird, muss das Fahrrad möglichst geradlinig auf Kurs gehalten werden. Dies geschieht mit dem Vorderrad, welches über den Lenker als Bedienoberfläche benutzerfreundlich angesteuert wird. Von der verbreiteten Maus und Menüsteuung ist das Fahrrad bisher verschont geblieben. Eine weitere wichtige Schnittstelle zwischen Mensch und Maschine ist der Sattel. Hier entscheidet sich auf mittlere Sicht das Schicksal des Verkehrsmittels und seines Operators. Ohne den optimal passenden Komfort macht es nun mal keinen Spaß.

Männer und Spielzeuge

Sowohl Computer als auch Fahrräder sind natürlich auch Spielzeuge. Prinzipiell könnten sie auch Spielzeug für Frauen sein, doch erfahrungsgemäß entwickeln Männer hier eine viel heftigere Leidenschaft. Der Preis von beiden hier behandelten technischen Objekten ist so unterschiedlich nicht, eine höherer Preis kann leicht das Prestige vermehren und das allseits beobachtete Erwachen der Sehnsucht nach weiterem Accessoire verhindert das schnelle Abflauen der Leidenschaft für die Spielgeräte. Empfohlen wird an dieser Stelle das Abonnement von Fachzeitschriften. Perfektion wird unermüdlich angestrebt und von der Werbung für die neue Produktgeneration versprochen. Bis zum eines nahen Tages fälligen radikalen Modellwechsel. Dann wird über Tod und Leben unserer technischen Lieblinge entschieden, denn die Update-Fähigkeit ist bald erschöpft, ein revolutionär gefederter Rahmen muss her oder eben unser Board hat jegliche industrielle Unterstützung längst verloren. Wenn man aber zu schnell umrüstet, ist man mit Sicherheit auch wieder bald der letzte.

Langlebigkeit und Bedienung

Bei 100 Jahre alten Fahrräder kann man Luft aufpumpen, den Rücken geradebiegen, losfahren und sich freuen über die Vollkommenheit und Solidität, die damals schon erreicht worden war. Falls man demgegenüber bei 20 Jahre alten Computern noch den On/Off-Schalter findet, wird man brutalstmöglich an der Bedienung scheitern. Doch auch die alten Computer wurden meist so gebaut, als müßten sie ewig halten. Sogar die Elektronik-Schrott Verordnung weiß nun nichts mehr mit ihnen anzufangen. Man wundert sich, wie konnte man mit den ollen Dingern so fabelhafte Sachen machen? Sogar schon Spiele.

Synthese

Sollen denn nun die Fahräder wie Computer gebaut werden oder gar die Computer wie Fahrräder? Der wahre Könner kann stets beides bedienen. Außerdem gibt es noch andere schöne Dinge, z.B. Eisenbahnen, Flugzeuge, Dampfer ....

Kontakt:

Martin Barghoorn, TU Berlin, Franklinstr. 28, D-10587 Berlin, eMail: Martin@Barghoorn.com

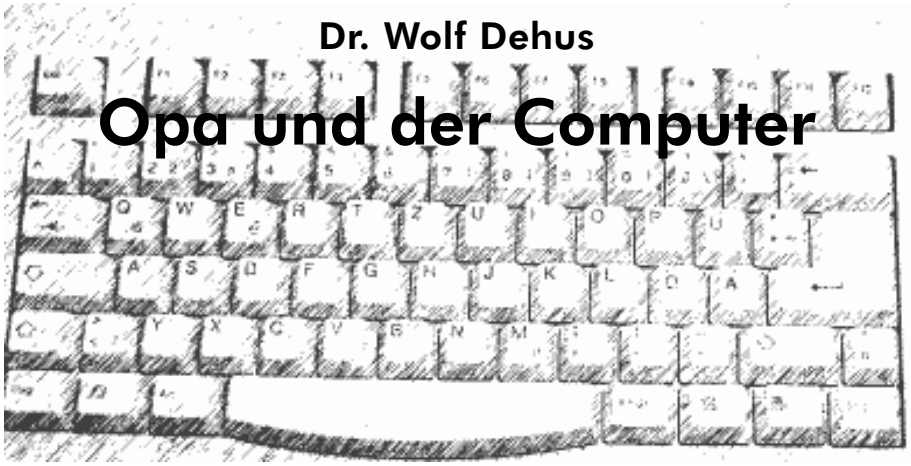
Vorschau:

Im nächsten APL-Journal lesen Sie unter anderem:

- **Beschreibung der ZUSE III** ***(Horst Zuse, Berlin)***
- **Die Fallstricke bei der Client/Server-Programmierung** ***(Conrad Hoesle-Kienzlen, Stuttgart)***
- **Lyx - open source document processor** ***(von Herbert Voß, Berlin)***
- **Durchdringung von Kugel und Zylinder** ***(von Diter Kilsch, Bingen)***
- **Opa und der neue Computer** ***(von Wolf Dehus, Berlin)***

Dr. Wolf Dehus

Opa und der Computer



Es ist schon einige Zeit vergangen, als uns unser gerade zehnjähriger Enkel wieder einmal besuchte, worüber wir uns stets freuten. Diesmal schaute er mich schon bald nach seiner Ankunft auf eine eigenartige Weise von oben bis unten an, und ich ahnte, daß – wie so manchmal – etwas Unerwartetes auf mich zukommt.

Und schon bald erklärte er mir, daß ich zwar sonst ganz in Ordnung wäre, aber trotzdem sei er von mir enttäuscht, weil ich noch nicht einmal einen Computer besitze. Natürlich ging das gegen meine Großvaterehre. Nach einigen für mich peinlichen Minuten fiel mir der Fußball ein, den ich ihm zwar schon vor längerer Zeit gekauft hatte, an diesem Tag aber mit einweihen durfte. Am anderen Morgen verfluchte ich diesen Kauf, denn ich konnte vor Bauchmuskelschmerzen kaum das Bett verlassen und mußte dann noch den Spott meiner häuslichen Regierung über mich ergehen lassen.

Als ich also den Fußball zur Sprache bringen wollte, ahnte mein Enkel schon, was kommt, und er unterbrach mich mit den Worten: „Aber Opa, das ist doch etwas ganz anderes!“ Dem konnte ich natürlich nicht widersprechen, zum einen, weil der Fußball wirklich etwas ganz anderes ist als ein Computer, und zum anderen, weil ich mich auch sonst mit meinem Enkel bestens verstand, vielleicht auch deshalb, weil ich in der Regel das tat, was er wollte.

Nun ging ich also auf die Suche nach einem Computer. Schon bald fand sich in der Nachbarschaft jemand, der bereit war, mir einen gebrauchten zu verkaufen. So wurde ich zum stolzen Besitzer eines Computers. Von den vielen Begriffen, die mir der Verkäufer nannte, habe ich nur die Zahl 286 behalten.

Am Abend rief ich meinen Enkel an, und fragte ihn, ob er Lust hätte, meinen neuen Computer mit den verschiedenen Geräten zusammenzubauen. Natürlich erklärte er sich sofort bereit, korrigierte mich aber gleich am Telefon mit den Worten, das seien keine Geräte, sondern die Hardware, wobei er mir gleich einige aufzählte. Darunter war sogar ein Tier, das mich erst in Schrecken versetzte, dann verließ ich mich aber auf meinen Enkel, daß dieses dazu gehöre wie z.B. der Hund zum Bauernhof.

Als er uns am nächsten Tag besuchte, fragte er gleich an der Wohnungstür, was das für ein Gerät sei. Ich erinnerte mich nur an die Zahl und erklärte geradezu stolz: „Das ist ein 286er“, denn das ist doch eine ziemlich hohe Zahl; weit höher als z.B. BMW 200.

Seine Antwort: „Opa, den hast du doch sicher nicht gekauft, sondern irgendwo bei der Müllabfuhr abgestaubt.“ Natürlich versuchte ich ihn von diesem Thema abzulenken und sagte, daß ich noch einige Platten dazu bekommen habe. Darauf er: „Aber Opa, das sind doch keine Platten, das sind Disketten.“ Zum Glück konzentrierte er sich mitten in dieser für mich unangenehmen Situation voll auf den Computer oder richtiger die vielen Kabel der Hardware und hatte nach und nach alles miteinander verbunden. Schon bald wirbelte er verschiedene Tafeln und Ähnliches auf die Bildfläche des Monitors. Ich war total sprachlos, weil ich staunte, was er alles aus dem kleinen Gerät mit einem so winzigen Volumen herausholte. Ich erinnere mich noch an ein Schachspiel, an verschiedene Inseln, die immer kleiner wurden, schließlich fand er auch noch Spielkarten mit der er eine Patience legte.

Dann wies er noch darauf hin, daß ich mit dem Computer rechnen und Briefe schreiben könne. Ich schwieg dazu und dachte nur, wie schön es wäre, wenn ich das wirklich könnte.

Nun wollte er plötzlich ein Fenster öffnen. Als ich ihn jedoch bat, das nicht zu tun, weil es schon kühl genug im Zimmer war, murmelte er nur noch etwas vor sich hin, was ich allerdings nicht verstand.

Kurze Zeit später besuchte uns mein Freund, der etwas von Computern verstand oder richtiger – auch etwas von Computern verstand. Denn in der Zwischenzeit hatte ich mir ein gewisses Computerwissen angeeignet und wußte bereits, daß eine Diskette nichts mit einem unmusikalischen Arm- oder Halsband zu tun hat, Online nicht mit einem Inlay zu verwechseln ist, Software kein Speiseeis und die Mouse ein technisches Gerät, Verzeihung, eine Hardware und kein Tier ist. Verstanden hatte ich auch, daß Enter nichts mit der Meerespiraterei zu tun hat.

Nun fragte ich meinen Freund, dem ich von meinem stolzen Erwerb erzählte, ob er mir ein paar Tips geben könne? Natürlich war er bereit, und schon am nächsten Tag saßen wir beide vor dem Computer. Mühsam hatte ich schließlich – wenn auch unvollständig – begriffen, wie ein Brief geschrieben wird und daß dieses Programm „Textverarbeitung“ heißt und vielleicht deshalb den so hochdeutschen Namen „Word“. trägt.

Als er mich erschöpft auf dem Stuhl mehr liegend als sitzend sah, beendete er die Computerlektion, vielleicht auch Doppel- oder Multilektion.

Dann fragte er aber doch noch, ob ein bestimmtes Spiel, dessen Name ich vergessen oder richtiger noch nicht behalten habe, im Computer eingespeichert sei, was ich sofort verneinte. Er mißtraute jedoch meinen Worten und setzte einige tastenähnliche Hebel oder hebelähnliche Tasten der Hardware in Bewegung; und da zeigte sich, daß in meinem Computer das Spiel doch eingespeichert war.

Ich kann mich nur noch erinnern, daß wir uns nun gegenseitig mit Bananen bewarfen; natürlich nicht mit richtigen! Trotz völliger Computererschöpfung riß ich mich zusammen, um meinem Enkel dieses Spiel zeigen zu können und notierte mir die einzelnen Buchstaben und Schritte, die zum Starten des Programmes notwendig waren. Zum Abschluß teilte mir mein Freund jedoch mit, daß ich dieses Spiel nur anwenden soll, wenn ich danach den Computer ausschalten möchte, da hinterher kein direkter Umstieg in ein anderes Programm möglich sei.

Als ich meinem Enkel beim nächsten Besuch dieses Spiel zeigen wollte, kam ich zwar einige Schritte voran, dann ging es aber nicht mehr weiter. Wie ich nun schwer enttäuscht und immer kleiner werdend vor dem Bildschirm saß, sagte mein Enkel belehrend: „Ja, Opa, wenn du nicht <ausführen> markierst, und mit Enter umsetzt, dann kann das ja auch nicht funktionieren!“ Schon bald hat er dann das Spiel einprogrammiert, und ich war ein fairer Verlierer.

Als ich ermüdet war, wollte er gern noch eine Runde Schach spielen. Ich gab ihm darauf den Hinweis meines Freundes, dessen Ratschlag ich noch im Ohr hatte, daß er den Computer ausschalten müsse und dann erst, also nach wiederholtem Einschalten des Gerätes, ein neues Programm eröffnen könne. Darauf lächelte er mich mit den Worten: „Ach, Opa!“ an und hatte bereits die Schachfiguren in Reih und Glied zu stehen. Da konnte ich nicht anders als ihn fragen, wie er das gemacht hatte. Darauf er: „Aber Opa, du weißt doch, daß man besonders das Kleingedruckte beachten muß. Ganz unten auf dem Bildschirm steht: „Zahl der Spiele“, wenn du dort nichts einstellst, gilt für die Spielzeit „unendlich“, und damit kommst du natürlich nicht aus dem Programm. Da hilft auch kein Enter oder Escape Ich glaubte ihm, auch ohne zu verstehen, was!

Als unser Enkel die Wohnung verlassen hatte, rief ich trotz meiner Computermüdigkeit meinen Freund an und belehrte ihn mit wortreichen und vielleicht auch etwas prahlerischen Sätzen über die Möglichkeit, vom Bananenspiel sofort in ein anderes Programm zu kommen. Natürlich verschwieg ich die Herkunft meines Wissens, um eigene Kreativität vorzutäuschen.

Nun fühlte ich mich schon fast wie ein Computerexperte, jedoch nur bis zum nächsten Besuch unseres Enkels, denn er brachte mich schon mit den ersten Worten schnell wieder auf den Boden der (Computer) – Realität zurück.

Brigitte Denck

Gott hat die Lausitz geschaffen, der Teufel hat die Kohle dort vergraben

(Sorbisches Sprichwort)

Kausche im Lausitzer Braunkohlerevier, eine Ortschaft, deren Geschichte über mehrere Jahrhunderte zurückverfolgt werden kann, mußte dem Tagebau weichen. Eine Serie von Farbfotografien, die die Künstlerin Brigitte Denck über den Ort erstellt hat, eingefasst in Originalfensterrahmen, wird im Folgenden mit ihren Titeln abgebildet. Fensterrahmen können die Zeitfenster symbolisieren. Verpackte und gekapselte vergangende Welten. Sie sind aus Materialien gestaltet, die Brigitte Denck auf ihren Spurensuchen aufgegriffen hat. Sie hat auch eine Tondokumentation erstellt, eindrucksvolle und monotone Geräusche aus dem nahen Tagebau begleiteten die Menschen in Kausche rund um die Uhr. Nach ihrer Rückkehr aus Südamerika wird Frau Denck ihre umfassende, datenbankgestützte Dokumentation vollenden, hier können nur einige wichtige Stücke vorgestellt werden.

Der Name Kausche kommt von der sorbischen Bezeichnung *„Chusej“*, übersetzt „Dorf im Besenginstergestrüpp“. Urkundlich wird der Ort erstmalig im Jahre 1527 erwähnt, er ist wahrscheinlich erheblich älter. Bei archäologischen Grabungen des Brandenburgischen Landesmuseums für Ur- und Frühgeschichte Potsdam, Arbeitsstelle Kausche, wurden Funde aus der Bronze- und Eisenzeit sowie aus der römischen Kaiserzeit (1500 v.Chr. bis 5. Jhd. n.Chr.) gemacht.

Die bis in unser Jahrhundert überlieferte Dorfstruktur des alten Dorfkerns ist seit dem 12. Jahrhundert weitgehend konstant geblieben. Bei Zerstörung, z.B. durch Brand, wurde immer wieder an der selben Stelle aufgebaut.

Um 1850 wurden erste umfangreiche Bohrungen zur Erkundung von Kohlelagerstätten durchgeführt. 1895 entstand die Kauscher Brikettfabrik und damit entwickelte sich Kausche zum grössten Ort der näheren Umgebung. Während des ersten Weltkrieges ließ der Einsatz von Frauen und Kriegsgefangenen die Zahl der dort Beschäftigten zeitweilig auf über 1.100 ansteigen.

Das Werk hat mehrere Generationen ernährt. Die Kohle gab Arbeit und Brot. Nun ist es wiederum die Kohle, der schließlich Kausche und Klein Görigk weichen müssen.

Im Dezember 1993 verpflichtete sich deshalb die Lausitzer Braunkohle AG, ein Dorf neu zu errichten und seine Bürger in unmittelbarer Nähe ihres bisherigen Heimatstandortes umzusiedeln (Kausche-Vertrag). Zum Ende des Jahres 1996 ist die Übersiedlung der rund 360 Einwohner beider Dörfer abgeschlossen. Der neue Ortsteil am Rande der Stadt Drebkau führt weiter den Namen „Kausche’.

Zwischen 1997 und 1998 besuchte ich die dem Abriss geweihte Ortschaft in regelmässigen Abständen. Ich fotografierte den langsam voranschreitenden Verfall und sammelte auf, was zurückgelassen wurde. Aus diesen Materialien entstand dann die Skulptur *Kausche*.

Die Fotografien habe ich in Originalfensterrahmen von Kausche eingefasst. So erhalten sie eine Aussagekraft, die die Vergangenheit in die Zukunft hinüberrettet.

Das Fenster zum Hof *(Brigitte Denck 1998)*

BRIGITTE DENCK

FENSTER

SKULPTUREN

INSTALLATIONEN

Noch besenrein übergeben und doch schon verfallen war Kausche, als Brigitte Denck im Juni 1997 zum ersten Mal den Ort im Lausitzer Braunkohlerevier aufsuchte. In den 60er Jahren wurde Kausche zum Braunkohlenrevier erklärt. Seitdem wussten die Bewohner: Eines Tages werden die Bagger den Ort schlucken. Viele zogen fort, andere blieben, reparierten das Notdürftigste, das drohende Ende stets im Hinterkopf. Schließlich siedelten die Kauscher um, der Ort blieb leer zurück und war doch voller Leben. Seit langem Aufgegebenes, Vergessenes entdeckte Brigitte Denck bei ihrem ersten Besuch in Kausche. Wie ein offenes Buch lagen die Gegenstände vor ihr. Ein Zeugnisbuch, Krankenliege, sogar ein alter Grabstein in der Scheune zeigten ihr die Geschichte des Dorfes und seiner Bewohner. Und ließen zugleich Raum, selber Geschichten um das Marode und die überwuchernde Natur zu spinnen. Geschichten um den Ort, aber auch um die eigene Geschichte, die Suche nach sich selbst. Regelmäßig fuhr sie fortan nach Kausche, stöberte durch die verlassenen Straßen und Häuser. Das Vergängliche, Zurückgelassene zog sie seit jeher an. Dinge, die achtlos weggeworfen scheinbar ihren Wert verlieren, oder neuen Wert bekommen, wenn man sie verändert.

Kontakt: brigitte.denck@gmx.de

Wintervorräte *(Brigitte Denck 1998)*

Zuhause - Blick aus dem Fenster *(Brigitte Denck 1998)*

Weitere Infos über Kausche:

www.berliner-journalisten-schule.de/depesche/kausche.htm

BRIGITTE DENCK

FENSTER
SKULPTUREN
INSTALLATIONEN

Wenn die Seele geht ... (Brigitte Denck 1998)

Gewitter (Brigitte Denck 1998)

Gebrochenes (Brigitte Denck 1998)

Erinnerung (Brigitte Denck 1998)

Die Stulle in der falschen Hand

Aus dem Tagebuch der Kindergärtnerin in Kausche

15.12.77
Steffen, Rene u. Sven verhalten sich beim Schlafen ganz ungezogen, Ingo M. u. Rene spucken.
Steffen B. führt ganz schlechte Reden.
Sven u. Steffen lachen.

10.12.
Rene sagt bei der Besprechung vor.
Steffen und Sven verhalten sich sehr schlecht bei der Besprechung.
Rene lacht trotz Ermahnung auf der Liege.
Thomas L., Steffen, Rene, Sven D., Ingo M., Ingo D. tumgetobt nicht aufgeräumt

20.12.
Steffen u. Sven toben vor dem Hinlegen trotz Ermahnung herum.
Rene zieht die Katrin, daß sie sehr hinfällt (aus reiner Boshaftigkeit)

6.12.79
Maik u. Andreas zerreißen eine hübsche Weihnachtsserviette.
Kathleen stößt ihre Brott. mit dem Fuß durch ganzen Gruppenraum.

13.12.
Heute haben David, Kathleen, Suse, Maik herumgetobt und nicht aufgeräumt.
Katrina wischt mit dem Finger auf der Erde herum u. schläft nicht. Maik u. Mireille sind immer ungezogen auf der Liege.

14.12.
Kathleen, Maik, Katrina, Mireille toben, anstatt aufzuräumen.
Maik, Mireille, David sind auf der Liege sehr ungezogen.

7.12.81
Mireille u. Manuela liegen nicht ordentlich auf der Liege.

9.12.
Manuela, Nicole, Sebastian u. Christian sitzen unter dem Tisch.

15.12.
Maik, Raik werfen die Bilderbücher auf den Fußboden.
Christian tobt im Kindergarten.

16.12.81
Mireille u. Maik sind nach dem Schlafen zu laut.

22.11.82
Die gesamte Gruppe wat vor der Beschäftigung ganz ungezogen.
Nach dem Mittagessen verhalten sich die Kinder trotz Ermahnung wieder ungezogen. Sie spielen mit dem Fleisch. Mireille wirft den Tischdienstlappen Steffen ins Gesicht.

6.12.82
Kinder waren nach dem Schlafen ungezogen.
Kinder träumen nicht ein.

7.12.82
Steffen, Thomas u. Silvio singen nicht gern Weihnachtslieder.
Mireille zieht dem Thomas den Stuhl weg.

16.12.
Frau S. konnte sich nicht auf die Gruppe verlassen.
Die Kinder waren beim Ausziehen ungezogen.

17.12.
Christian, Raik u. Mario sind im Wald weggelaufen
Alle Kinder der großen Gruppe essen die Teller sauber ab.

27.11.84
Henry S. war beim Mittagessen u. vor dem Schlafengehen sehr ungezogen.
Oliver u. Silvio ärgern die Kinder.
Christian W. spielt mit dem Gebauten.

Obwohl Christian im Buch steht ist er sehr ungezogenauf der Liege.

27.11.
Silvio, Nico, Henry u. Mario hören nicht beim Spatzierengehen.

3.12.
Henry u. Christian toben herum.

4.12.
Raik u. Christian fassen sich nicht im Morgenkreis mit einem anderen K. an. Torsten schwatzt beim Frühstück.
Christian spricht auch.
Torsten schwatzt trotzErmahnung.
Torsten, Silvio u. Nico sind beim Turnen ungezogen.

5.12.84
Oliver spuckt über den Tisch.

7.12.
Oliver ist sehr frech. Silvio macht Oliver alles nach. Henry S. ist sehr laut auf der Liege.

10.12.
Christian stößt Danny mit dem Stuhl das gleich die Lippe blutet.
Bianka, Raik u. Denny toben im Gruppenraum, hören nicht, müssen sich auf den Stuhl setzen

10.12.
Silvio P. wirft die Fußmatten durch die Luft und versteckt die Fußmatten
Frau K. sagt aufräumen und Danny antwortet: wir räumen nicht auf.
Henry S. trotz Verbot geht er wieder hinten aufs Eis.
Oliver schubst Jaquline vom Fenster.

30.11.85
Nico, Mario u. Henry toben beim Spaziergang.

5.12.85
Anett D. ist nicht ruhig beim Schlafen.

10.12.85
Christian, Stephanie, Annette B., Mario, Daniel haben immer die Stulle in der falschen Hand, trotz Ermahnung.
Jaqueline, Nico u. Ralf sitzen nie richtig auf dem Stuhl

6.12.85
Christian hat den Mario 2 x sehr verletzt.

9.12.85
Alle Kinder werden im Schlafraum laut.

10.12.85
Nico u. Mario räumen nicht auf. Christian u. Silvio pantschen in der Bauecke.

(Auszüge aus dem Tagebuch der Kindergärtnerin in Kausche)

Altes DDR-Plakat, gefunden in Kausche
Foto: Brigitte Denck

Allgemeine Informationen

(Stand 1. Dezember 1999)

Vorstand

Vorsitzender:

Dieter Lattermann
Rheinstraße 23
69190 Walldorf
Tel. (06227) 63469
E-Mail: dieter_lattermann@compuserve com

2. Vorsitzender:

Martin Barghoorn
Technische Universität Berlin
Sekr. FR 6-9
Franklinstr. 28
10587 Berlin
Tel. (030) 314 24392
Fax: (030) 314 25901
E-Mail: barg@cs.tu-berlin.de
Web: http://stat.cs.tu-berlin.de/~barg/

Schriftführer

Wolfgang Dreßen
McKinsey & Co.
Königsallee 60c
40027 Düsseldorf
Tel. (0211) 136-443
FAX: 0211/136-4997-4437
E-MAIL: wolfgang_dressen@McKinsey.com

Schatzmeister

Gert Osterburg
Am Hochholz 7
61184 Karben
Tel. (06034) 2995
E-Mail: Gert.Osterburg@T-Online.de

Beitragssätze

Ordentliche Mitglieder:

Natürliche Personen	60,00	DM	p.a.
Studenten / Schüler	20,00	DM	p.a.

Außerordentliche Mitglieder:

Juristische / natürliche Personen	≥	1.000,00	DM	p.a.

Bankverbindung: BVB Volksbank eG Bad Vilbel (BLZ 518 613 25), Konto-Nr. 523 2694

Hinweis:

Wir bitten alle Mitglieder, uns immer gleich *Adressenänderungen* und *neue Bankverbindungen* mitzuteilen.

Geben Sie bei *Überweisungen* den *Namen* und/oder die *Mitgliedsnummer* an.

Einzugsermächtigung

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V.

den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht.

Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____

APL Germany e.V. Aufnahmeantrag/Änderungsanzeige

APL Germany e.V.
Herrn Gert Osterburg
Am Hochholz 7

D-61184 Karben

❑ Ich ersuche / wir ersuchen um Aufnahme in den *APL Germany e.V.*

als außerordentliches Mitglied (❑ jur. Person ❑ natürl. Person)

Die Vereinssatzung ist mir / uns bekannt: Ja ❑ Nein ❑

❑ Änderungsanzeige

Der Verein darf Aufkleber mit meiner / unserer Postanschrift weitergeben:

Ja ❑ Nein ❑

	Dienstanschrift/ Hauptansprechpartner	Gegebenenfalls weitere Mitglieder der Organisation
Firma		
Abteilung o.ä.		
zu Hd. von		
Straße, Hausnummer		
Postfach		
PLZ, Ort		
Bei Ausland: Staat		
Telefon (Vorwahl / Ruf)		
FAX (Vorwahl / Ruf)		
E-Mail-Adresse		

Die Angaben dieses Formulars werden für vereinsinterne Zwecke auf Datenträger gespeichert.

Ort, Datum: _____ Unterschrift: _____

APL2001
An Arrays Odyssey

June 25-28, 2001
Yale University, New Haven, CT, U.S.A.

ACM/SIGAPL is pleased to announce that APL2001, the annual international conference on APL and other array programming languages, such as A, A+, J, K, Matlab, Mathematica, Nial, and S-Plus, will be held June 25-28, 2001. The theme is An Arrays Odyssey to celebrate our launch into the multi-dimensional space of the new millennium. An international committee headed by NY/SIGAPL is planning the conference.

Conference Site | Schedule | Speakers | Papers | Tutorials | Vendors | Tours | Travel | Hotels | Tourism | Registration

http://www.apl2001.org/

http://www.apl2001.org/site.htm

APL2001
An Arrays Odyssey

Conference Site

Home | Conference Site | Schedule | Speakers | Papers | Tutorials | Vendors | Tours | Travel | Hotels | Tourism | Registration

The conference will be held at Yale University, which is celebrating its 300th anniversary. Yale has made important contributions to the development of APL, both directly and indirectly through its graduates and faculty. We hope to make the conference compact and affordable. Lodging will be available in new high quality student housing where up to 180 of our participants will be able to stay. Rooms will also be reserved at local hotels. Meals will be held in one of the college dining halls. Presentations will be made in the facilities of the department of engineering.

New Haven, Connecticut is located on Long Island Sound within an hour and a half from major international airports in New York City and Hartford, CT. Train and limo service is available. The University is a short taxi ride from the train station and limo depot. Besides Yale's excellent museums, there are a number of tourist sites ranging from dinosaur footprints to colonial whaling ships within easy reach.

Yale University
Yale University Map
Conference Location Map

Conference will take place in Davies Hall, south of Becton Hall on Prospect Street.

---

http://www.vector.org.uk/

How to use this site | About APL | APL Fonts | BAA News | Subscribing to Vector | Products | Events | [More...]

Out Now! 17:4...
includes reports and pictures of
Finnish Forests V

Spot The Axe Competition
Last chance to enter

Notice of BAA AGM and Vendor Forum 18th May
News of APL2001 in Yale 25-28 June 2001

VECTOR

Vol 17 No.4
April 2001

The Journal of the British APL Association

More eXtreme APL...

Contributing to Vector | Order form for back issues | Resources | Advertising | Contacts | Jobs

To receive an automatic e-mail when this page is updated, please enter your e-mail address: Submit

Webmaster: Ray_Cannon@compuserve.com
Vector Administration: Gil Smith at apl385@compuserve.com

http://www.torontoapl.org/

The Toronto *APL* Special Interest Group

Welcome to the Toronto *APL* SIG! The activities of Toronto members are posted here to promote sharing with the rest of the *APL* community.
"*APL*" is both *A Programming Language*, and an acronym for *Array Processing Languages*, such as *APL*, A, A+, J, K, Matlab, Mathematica, Nial, and S-Plus.

Next Meeting

To Be Announced

Previous Meetings - and Contact Info

Other Info

The *APL* Skills Database
Employment Information for *APL* companies and *APL* professionals

Get Connected! - To Our Automated Mailing List
Read this info, and Subscribe Yourself to our new automated mailing list.
Don't miss out on announcements for upcoming events and meetings, conference information, software vendor forums, and more.

The *APL97* Art Gallery "Chaos with Symmetry"

More Info:

- The *APL* FAQ
- *APL* FAQ - alternate posting
- Previous Meetings, Meeting Summaries, etc. - read some reviews about our past meetings here