APL-Journal

A Programming Language

1/2002

APL-Germany e.V.

Jahrgang 21 Nr. 1 2002

ISSN - 1438-4531

RHOMBOS-VERLAG

In dieser Ausgabe:

Wolfgang Strasser
Miszellen aus der Finanzmathematik

Kai Jäger
Dyalog APL-Namespaces++: DCOs

Nancy Wheeler
APL2 Namespaces

Im Blickpunkt

Peter-Michael Hager
Darstellung von ASN.1 in genesteten APL Strukturen

Inhaltsverzeichnis

Peter-Michael Hager
Darstellung von ASN.1 in genesteten APL Strukturen 4

Nancy Wheeler
APL2 Namespaces
An Inside Look at the Journey from Mainframe to Workstation 25

Kai Jäger
Dyalog APL-Namespaces++:
Dyalog Component Objects (DCOs) 27

Wolfgang Strasser
Miszellen aus der Finanzmathematik 33

Presentation:
Workstation APL2 for Multiplatforms Version 2.0 41

Aufnahmeformular 44

Impressum 2

Editorial 3

Vorschau

Im nächsten APL-Journal lesen Sie unter anderem:

David Libtag (IBM)

About APL2 Gui Tools

Dr. Rainer Nußbaum

Automation für Linux und Unix

Dr. Werner Schäfer

Verbesserung der Genauigkeit bei der numerischen Lösung von linearen Differentialgleichungen

4 Darstellung von ASN.1 in genesteten APL Strukturen

Abstract Syntax Notation One, ASN.1, definiert von der ITU, ist ein standardisiertes Datenformat für Übermittlungen in Netzwerken und zur Kommunikation in offenen Systemen. Diverse TCP/IP Protokolle und Sicherheits-Anwendungen basieren darauf. Um APL zur Realisierung von Software-Lösungen in diesen Feldern einsetzen zu können, benötigt man Hilfsmittel, mit denen Daten im ASN.1-Format unter APL gehandhabt werden können. Dieser Artikel verschafft einen Überblick über Notation und Encoding in ASN.1, und beschreibt ein Verfahren, mit dem komplexe ASN.1 Inhalte in genestete APL Strukturen transoformiert werden können und umgekehrt. Des Weiteren wird die Handhabung von Crypto-Zertifikaten unter Dyalog APL behandelt. Es ist sogar möglich, APL dahingehend zu erweitern, dass jedes APL-Objekt, bis hin zu einem kompletten Arbeitsbereich, nach ASN.1 konvertiert werden kann.

25 APL2 Namespaces: An Inside Look

This article describes how you get one step closer to true cross-system compatibility. The „Journey from Mainframe to Workstation“ leads to a working implementation for workstation APL2, close adherence to the existing design, full compatibility at execution time, minor differences in namespace creation, minor differences in invocation options and a few improvements over the mainframe implementation (i.e. variables not copied to active workspace and more flexibility in namespace creation).

27 Dyalog APL-Namespaces++: Dyalog Component Objects

Mit Hilfe von Namespaces können Code wie Variable gekapselt und zugleich strukturiert werden. Dies führt zu einem Code, der leichter zu erstellen, zu verstehen und zu warten ist. Eigens für die Unterstützung von größeren Programmierteams wird eine Konvention vorgestellt, die definiert, welche Informationen ein Namespace enthalten muss oder sollte. Außerdem wurde Software für die Verwaltung solcher Namespaces realisiert, mit der auch größere Programmierteams unterstützt werden. Namespaces, die diese Konventionen einhalten, werden als DCOs („Dyalog Component Objects“) bezeichnet.

33 Miszellen aus der Finanzmathematik

Der Autor erörtert verschiedene Ansätze der Finanzmathematik, mit denen die zukünftige Entwicklung eines Aktienportfolios anhand der Vergangenheitsdaten abgeschätzt werden soll.

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,
Internet: www.rhombos.de

Gestaltung: RHOMBOS-VERLAG, Berlin

Druck: dbusiness.de GmbH, Berlin

Copyright: *APL Germany e.V.*
(für alle Beiträge, die als Erstveröffentlichung erscheinen)

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

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung von Redaktion und Herausgeber nicht übernommen werden. Sämtliche Veröffentlichungen im APL-Journal erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

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 schwarz-weiß 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, September 2002

Liebe APL-Freunde,

eigentlich hätten Sie dieses *APL-Journal* kurz vor der Konferenz APL 2002 in Madrid (22. bis 25. Juli 2002) in der Hand haben sollen. Leider haben wir das nicht geschafft, wir bitten um Nachsicht und geloben Besserung.

Zunächst sei den Autoren, die ganz freiwillig oder sanft gedrängt ihre Beiträge abgeliefert haben, herzlich gedankt. Besondere Dank gebührt wieder unserem Editor, der nicht locker gelassen hat, potenzielle Schreiber zu realen zu machen.

Wie seit Langem üblich, bieten die Tagungen von *APL Germany* ein Forum zur Präsentation neuer Entwicklungen der APL-Anbieter. Auf der letzten Tagung in München, die in den Räumen der IBM statt fand, konnten wir erfahren, dass auch IBM auf die Zukunft von APL setzt und ein neues Produkt ankündigt. Einzelheiten können Sie in diesem Heft nachlesen. Hervor gehoben verdient schon hier, dass die neue Version von APL2 nicht länger eigenen Wege bei der Gestaltung von Oberflächen geht, sondern Schnittstellen zu bewährten oder zumindest weit verbreiteten GUI-Werkzeugen wie Tcl anbietet.

Unser Mitglied Tilman Otto, der in seinem APL2C („APL-to-C“ für Nicht-Anglophile) den Grundstein für die Verwendung von Tcl für APL-Anwendungen gelegt hat, wird im nächsten *Journal* über diese Werkzeuge informieren.

Nicht untätig geblieben bei der Weiterentwicklung ihrer APL-Systeme sind auch alle anderen Anbieter: Dyadic Systems wird über Neuigkeiten in Dyalog APL berichten. APL2000 hält eine eigene Konferenz, die sich leider mit unserem Treffen im November zeitlich überlappt. Micro-APL ist auf den Markt zurückgekehrt, wir werden die Firma zu einer späteren Tagung einladen.

Die ursprüngliche Idee von APL – die konzise und sehr kompakte Formulierung von Algorithmen – ist in der Klick-Welt vielleicht etwas aus der Mode geraten. Ich höre hauptsächlich von APL-Pensionären, dass die sich mit solchen Fragestellungen befassen. Die kommerzielle Nutzung der algorithmischen Vorteile findet zwar nach wie vor, aber weitgehend unter Ausschluss der Öffentlichkeit, statt. Deshalb möchte ich unsere APL-Veteranen herzlich einladen, über ihre „Spielereien“ mit APL für das *Journal* zu berichten. Haben Sie keine Scheu, scheinbar Triviales zu beschreiben, vielleicht liest das ja auch einmal ein Nicht-APLer, der sehr überrascht sein wird. Und die alten Hasen werden sich freuen, wieder einmal APL-Code zu sehen.

Einen Bericht über die Madrid-Konferenz werden Sie im *Journal* vergeblich suchen – wir verweisen auf den „Vector“ und das Internet. Nur so viel sei gesagt: Madrid war den meisten APLern zu weit, nur 88 Teilnehmer fanden den Weg dorthin.

Ich wünsche Ihnen viel Freude bei der Lektüre dieses *Journals* und grüße Sie herzlich,

Ihr

Dieter Lattermann

Peter-Michael Hager

Darstellung von ASN.1 in genesteten APL Strukturen

Abstract Syntax Notation One, ASN.1, definiert von der ITU, ist ein standardisiertes Datenformat für Übermittlungen in Netzwerken und zur Kommunikation in offenen Systemen. Diverse TCP/IP Protokolle und Sicherheits-Anwendungen basieren darauf. Um APL zur Realisierung von Software-Lösungen in diesen Feldern einsetzen zu können, benötigt man Hilfsmittel, mit denen Daten im ASN.1-Format unter APL gehandhabt werden können. Dieser Artikel verschafft einen Überblick über Notation und Encoding in ASN.1, und beschreibt ein Verfahren, mit dem komplexe ASN.1 Inhalte in genestete APL Strukturen transoformiert werden können und umgekehrt. Des Weiteren wird die Handhabung von Crypto-Zertifikaten unter Dyalog APL behandelt. Es ist sogar möglich, APL dahingehend zu erweitern, dass jedes APL-Objekt, bis hin zu einem kompletten Arbeitsbereich, nach ASN.1 konvertiert werden kann.

Seit 17 Jahren ist ASN.1 der Standard zur Beschreibung von abstrakten Objekten im Open Systems Interconnection Modell (OSI, definiert in ITU X.200). ASN.1 ist als Daten-Beschreibungssprache weit verbreitet in Kommunikationsprotokollen wie dem Simple Network Management Protocol (SNMP), aber auch in den digitalen Zertifikaten nach ITU X.509. Die Zertifikats-Infrastruktur für digitale Signaturen, Zeitstempel, Trustcenter, Authentisierung, Codesignatur, S/MIME, Verschlüsselung, Sperrlisten und die Public-Key Cryptographic Standards (PKCS) der RSA sind alle vollständig in ASN.1 formuliert. Jegliche strukturierbare Daten können in der ASN.1-Notation syntaktisch beschrieben werden. Im OSI Model behandelt ASN.1 den Presentation, und teilweise auch den Application Layer (siehe Tabelle). ASN.1 besteht aus zwei Arten von Basis-ASN.1 besteht aus zwei Arten von Basis-Deklarationen, der Spezifikation der Syntax-Notation (ITU X.208/X.680), und der Spezifikation der Encoding-Vorschriften, für die Basic und Distinguished Encoding Rules – BER und DER, in der ITU X.209/X.690 definiert.

Layer	Beschreibung
7	Application Layer
6	Presentation Layer
5	Session Layer
4	Transport Layer
3	Network Layer
2	Data Link Layer
1	Physical Layer

Ein Notations-Beispiel

```
-- Kommentare beginnen mit doppeltem Bindestrich und enden am Zeilenende oder beim nächsten —
-- Es ist bei Spezifikationen zur Konvention geworden, Notationen in Anhang A zusammenzufassen.

MyLetterFramework        -- Dieses ist der identifizierende Name des Moduls
DEFINITIONS ::=
BEGIN                    -- Hier fangen die Zuweisungen für das Modul an

-- Unter ASN.1 könnte ein Brief etwa so aussehen:

Letter ::=      SEQUENCE {          -- Brief ist eine Sequenz von:
     sender          Address,       --     Absender,
     addressee       Address,       --     Empfänger,
     message         T61STRING,     --     Nachricht,
     date            UTCTIME }      --     und Datum.

-- SEQUENCE, T61STRING und UTCTIME sind Basisdatentypen,

-- die ASN.1 von Hause aus schon kennt,
-- Address ist ein Anwender-definierter Typ, hier wie folgt definiert:
```

```
Address ::=          SEQUENCE {          -- Adresse ist eine Sequenz von:
    name             PRINTABLESTR,       --     Name,
    street           PRINTABLESTR,       --     Straße,
    town             PRINTABLESTR }      --     und Ort.

-- PRINTABLESTR ist wiederum ein Basisdatentyp.


END                  -- Hier enden die Zuweisungen für das Modul


-- Die vollständig auf Basisdatentypen expandierte Definition unseres
-- Briefes Letter sieht dann so aus:
--
-- Letter ::=        SEQUENCE {
--   sender          SEQUENCE {
--       name            PRINTABLESTR,
--       street          PRINTABLESTR,
--       town            PRINTABLESTR },
--   addressee       SEQUENCE {
--       name            PRINTABLESTR,
--       street          PRINTABLESTR,
--       town            PRINTABLESTR },
--   message         T61STRING,
--   date            UTCTIME }
```

Die ASN.1 Notation

Die Beschreibung von Objekt-Strukturen erfolgt in ASN.1 durch das Definieren von Modulen, von denen jedes durch einen derartigen Block dargestellt wird: DEFINITIONS, BEGIN, EXPORTS, ALL und END sind Beispiele für reservierte Schlüsselworte aus der ITU X.208/X.680.

Notation der Modul-Definition	Beispiel
<Module Identifier>	MyLetterFramework
DEFINITIONS ::=	DEFINITIONS ::=
BEGIN	BEGIN
<Parameterized Definitions>	EXPORTS ALL
<Module Body>	<Modul-Inhalt>
END	END

Im Module Body werden die Objekte und Werte in Form von Zuweisungen, falls erforderlich, auch in genesteter Form, deklariert: SEQUENCE und INTEGER sind Beispiele für vordefinierte Datentypen der Klasse UNIVERSAL, ebenfalls aus der ITU X.208/X.680.

Notation des Modul-Inhalts	Beispiel
<Module Body>	`Letter` ::= SEQUENCE {..}
<Assignment-1>	**Address** ::= SEQUENCE {..}
<Assignment-2>	**MyVersion** ::= INTEGER (2)
... oder allgemein …	...
<Assignment-n>	<Type Reference> ::= **<Type>**

Andere Module können über Import- und Export-Parameter auf unseren `Letter` verweisen: IMPORTS, FROM und SET OF sind weitere reservierte Schlüsselworte aus der ITU X.208/X.680.

Notation der Modul-Definition	Beispiel
<Module Identifier>	MyContainerFramework
DEFINITIONS ::=	DEFINITIONS ::=
BEGIN	BEGIN
<Parameterized Definitions>	IMPORTS Letter FROM MyLetterFramework
<Module Body>	`Container` ::= SET OF `Letter`
END	END

Typen

Jeder **<Type>** Ausdruck wird durch drei Komponenten bestimmt: *Class Form Tag*. Innerhalb einer *Class* spezifiziert *Tag* den Datentyp. Diese vier Klassen *Class*, kennt ASN.1 (hier erweitert um das Präfix **CLASS_**):

```
CLASS_UNIVERSAL←0        ⍝ Wird ausschließlich in der ITU X.208/X.680 spezifiziert
CLASS_APPLICATION←1      ⍝ Typ-Zuweisungen anderer Spezifikationen (selten verwendet)
CLASS_PRIVATE←3          ⍝ Firmenspezifisch, keine ITU-Belegung (selten verwendet)
CLASS_CONTEXT←2          ⍝ Bedeutung abhängig vom Kontext der Verwendung
```

Die *Tag*-Nummer ist eine nicht negative ganze Zahl, deren Bedeutung von der Klasse *Class* abhängt, in der sie spezifiziert ist. Die Basisdatentypen, die ASN.1 aus der ITU X.208/X.680-Spezifikation bereits mitbringt, sind in der *Class* UNIVERSAL angesiedelt. Dieses sind, nach Kategorien eingeteilt, die vordefinierten *Tags* in dieser Klasse:

UNIVERSAL 1	Boolescher Typ
UNIVERSAL 2	Integer Typ
UNIVERSAL 3-4	String Typen
UNIVERSAL 5	Null Typ (keine Daten)
UNIVERSAL 6-8	Objekt kennzeichnende Typen
UNIVERSAL 9	Real Typ
UNIVERSAL 10	Aufzählungs-Typ
UNIVERSAL 11	Gekapselter Darstellungs-Typ
UNIVERSAL 13-15	Reserviert für zukünftige Erweiterungen
UNIVERSAL 16	Sequence und Sequence-of Typen
UNIVERSAL 17	Set und Set-of Typen
UNIVERSAL 12,18-22	Zeichenketten Typen
UNIVERSAL 23-24	Datums-Typen
UNIVERSAL 25-30	Zeichenketten Typen
UNIVERSAL 31-..	Reserviert für Nachträge zur X.208/X.680

In anderen ITU-Spezifikationen, als der X.208/X.680 wird *Class* APPLICATION benutzt, um anwendungsspezifische *Tag*-Nummern zu vergeben. Dort wird das Schlüsselwort APPLICATION in Verbindung mit der *Tag*-Nummer innerhalb dieser Klasse verwendet:

```
APPLICATION 50        SpecialX9999Type
```

Für eigene Datentypen wird *Class* PRIVATE verwendet. Die Notwendigkeit, dazu dürfte jedoch eher selten sein, da sich das meiste mit den Typen der ITU ausdrücken lässt:

```
PRIVATE 1             MyTriStateType
```

Bleibt noch die *Class* CONTEXT, um bei Bedarf kontextspezifische Erweiterungen in einer Anwendung zu markieren. Hierbei werden die *Tag*-Nummern oftmals in Abhängigkeit von der Bedeutung des explizit enthaltenen Daten-Typs vergeben. In einem späteren Teil, der die Struktur eines X.509 Zertifikates beschreibt, werden Beispiele für die Verwendung von *Class* CONTEXT gegeben. Im Gegensatz zu APPLICATION oder PRIVATE wird hierbei nicht das Schlüsselwort CONTEXT mit nachfolgender *Tag*-Nummer verwendet. Stattdessen wird die *Tag*-Nummer in eckigen Klammern angegeben:

```
[0]        INTEGER {0,1,2} DEFAULT 0
[3]        Extensions OPTIONAL
```

Hinweis:

DEFAULT bedeutet, dass der gesamte Eintrag weggelassen werden sollte, wenn der INTEGER, der zum Beispiel eine Versionsnummer angeben könnte, 0 ist.

OPTIONAL bedeutet, dass „Extensions" nicht notwendigerweise verwendet werden muß.

In der ASN.1 Syntax-Notation gibt es noch weitere Schlüsselworte, die in den ITU Spezifikationen erläutert sind.

Als drittes Merkmal neben *Class* und *Tag*-Nummer besitzt jeder Type noch ein *Form*-Attribut, das ihn entweder als PRIMITIVE oder CONSTRUCTED kennzeichnet. Der Dateninhalt der primitiven besteht aus einem der Basistypen, wohingegen bei den konstruierten Typen der Dateninhalt wieder einen, oder mehrere aneinandergereihte ASN.1-Typen enthält. Die Schlüsselworte dazu sind (hier erweitert um das Präfix **FORM_**):

```
FORM_PRIMITIVE←0        ⍝ Primitive     (z.B. bei PRINTABLESTR oder UTCTIME)
FORM_CONSTRUCTED←1      ⍝ Constructed (z.B. bei SEQUENCE oder SET)
```

Anzumerken bleibt, dass in der Praxis leider nicht überall PRIMITIVE drin ist, wo es darauf steht. Die String-Typen, Bitstring und Oktettstring beinhalten oftmals ASN.1-Encodings, was damit zu erklären ist, dass etwa die ITU X.509 den SubjectPublicKey eines Zertifikats neutral als Bitstring festlegt. Dieser ist jedoch meist ein RSA-Schlüssel, der von der RSA in PKCS#1 als ASN.1-SEQUENCE definiert wurde.

Bei Encoding stellt dieses noch kein Problem dar, sofern die Encoding-Software tolerant genug ist, als Daten-Wert eines Bitstrings auch eine byteorientierte ASN.1-Sequenz zu akzeptieren. Um beim Decoding nicht nur den Bitstring zu erhalten, sondern den darin versteckten RSA-Schlüssel wieder zu gewinnen, muss dieser Bitstring entweder erneut decodiert werden, oder es wird ein sogenanntes spekulatives Decoding auf diese String-Typen angewendet. Beim spekulativen Decoding sucht die Software die Inhalte dieser String-Typen mit hoch kritischen Regeln nach vollständig gültigem ASN.1-Code ab. Bei Erfolg liefert sie dann den Inhalt decodiert zurück, andernfalls bleibt es ein String-Typ.

Dieses Verfahren kann problematisch werden, wenn aus einem Bitstring, der wirklich nur ein Bitstring war, plötzlich eine ASN.1-Konstruktion interpretiert wird.

In der Praxis ist mir allerdings ein derartiger Fall noch nicht begegnet, da die Wahrscheinlichkeit der Fehlinterpretation weit geringer als sechs Richtige im Lotto ist.

Die vordefinierten Tags

Sämtliche in der X.208/X.680 vordefinierten Tags gehören der *Class* UNIVERSAL an und werden durch ihre *Tag*-Nummer unterschieden. Sie werden üblicherweise durch reservierte Namen referenziert, statt durch das Schlüsselwort UNIVERSAL und die *Tag*-Nummer. Die wichtigsten sind (hier erweitert um das Präfix **TAG_**) :

```
TAG_BOOLEAN←1                ⍝ TRUE oder FALSE
TAG_INTEGER←2                ⍝ Einzelner Integer beliebiger Genauigkeit
TAG_BITSTRING←3              ⍝ Bit-Sequenz
TAG_OCTETSTRING←4            ⍝ Byte(Oktett)-Sequenz
TAG_NULLTAG←5                ⍝ NULL (Keine Daten)
TAG_OID←6                    ⍝ Object Identifier (Numerische Identifikations-Sequenz)
TAG_REAL←9                   ⍝ Real (Mantisse × Basis Exponent)
TAG_ENUMERATED←10            ⍝ Numerierung (Index)
TAG_UTF8STR←12               ⍝ UTF-8 Zeichenkette
TAG_SEQUENCE←16              ⍝ Constructed Sequence / Sequence Of
TAG_SET←17                   ⍝ Constructed Set / Set Of (Satz ungeordneter Instanzen)
TAG_NUMERICSTR←18            ⍝ Numerischer Zeichenstring (Ziffernfolge)
TAG_PRINTABLESTR←19          ⍝ Druckbarer String (Beliebige Anzahl druckbarer Zeichen)
TAG_T61STRING←20             ⍝ Teletext String
TAG_IA5STR←22                ⍝ IA5 String (Beliebige Anzahl von ASCII Zeichen)
TAG_UTCTIME←23               ⍝ UTC Time (1950-2049 mit zweistelliger Jahreszahl)
TAG_GENERALIZEDTIME←24       ⍝ Generalized Time (vierstellige Jahreszahl)
TAG_GENERALSTR←27            ⍝ Zeichen-String allgemein
TAG_UNIVERSALSTR←28          ⍝ Unicode (4 Byte Universal)
TAG_BMPSTR←30                ⍝ Basic Multilingual Plane, 2 Byte Unicode-Untermenge
```

Einer der bedeutendsten Tags ist die SEQUENCE (UNIVERSAL 16) Konstruktion. Sie wird benutzt um genestete, oder in der ITU-Terminologie, „CONSTRUCTED Encodings“ zu beschreiben.

Hiermit wird ASN.1 in die Lage versetzt, Strukturen beliebiger Tiefe zu definieren. Eine SEQUENCE kann man sich als Ansammlung von ASN.1-Elementen vorstellen, wobei jedes Element von beliebigem Typ sein darf. Die Reihenfolge der Elemente ist bei SEQUENCE maßgeblich.

CONSTRUCTED Encodings können des weiteren mit dem SET (UNIVERSAL 16) Typ gebildet werden. Genauso wie SEQUENCE kann auch SET eine beliebige Anzahl von Elementen beliebiger Typen beinhalten. Aber im Unterschied zu SEQUENCE besteht SET aus einer nicht geordneten Ansammlung von Elementen.

Bei SEQUENCE und SET werden die Type-Definitionen ihrer Elemente üblicherweise einzeln vorgegeben.

Als weitere Variante zu SEQUENCE and SET gibt es noch SEQUENCE OF bzw. SET OF. In diesen Strukturen dürfen sich die Elemente beliebig oft, oder auch gar nicht, wiederholen. Unterschieden werden diese Varianten durch die Syntax-Definition und ihre Interpretation. Ihr Encoding ist identisch.

Das Encoding von Daten

Die Form, in der ASN.1-notierte Daten vorliegen, ist nicht die für Menschen lesbare, bislang vorgestellte, sondern eine verdichtete oder encodierte Form, die sich zur Kommunikation zwischen Maschinen besser eignet. Jedes Informations-Element besteht aus einem Header, gefolgt von dem jeweiligen Daten-Wert. Der Header beschreibt die Eigenschaften der Struktur. Diese besteht aus:

- **Class**
- **Form**
- **Tag** und
- **Length** des folgenden Daten-Werts.

i Hinweis:

Die Eigenschaft Class wird auf zwei Bit encodiert: UNIVERSAL (00),APPLICATION (01), CONTEXT (10) oder PRIVATE (11). Die Eigenschaft Form wird auf einem Bit encodiert: PRIMITIVE (0) oder CONSTRUCTED (1).

Primitive bedeutet, dass der Daten-Wert direkt auf den Header folgt, Constructed, dass die dem Header folgenden Daten wiederum aus Header und Daten-Wert Anreihungen bestehen. Die Eigenschaft Tag ist die zuvor erwähnte Nummer innerhalb der Class. Die Eigenschaft Length des Inhalts, als letzter Bestandteil des Headers, gibt die Anzahl Oktette (Bytes) an, die dem Header folgen.

Für eine kompakte Darstellung des Headers sind die drei ersten Bestandteile in einem einzigen *Identifier* Oktett untergebracht, sofern der *Tag*-Nummer kleiner als 31 ist. Die Bits werden von rechts nach links nummeriert:

8	7	6	5	4	3	2	1
Class		Form	Tag				

Für *Tag*-Nummern, die größer oder gleich 31 sind, besteht das Identifier Encoding aus zwei oder mehr Oktetten. Das abschließende Oktett wird durch ein auf Null gesetztes Bit 8 identifiziert:

8	7	6	5	4	3	2	1
Class		Form	1	1	1	1	1

8	7	6	5	4	3	2	1
1	Höherwertige Bits von Tag (Mehrere Oktette möglich)						

8	7	6	5	4	3	2	1
0	Niederwertigste Bits von Tag						

Direkt auf die Identifier Oktette folgen die *Length* Oktette. Auch diese sind in einer kurzen, sowie einer langen Form definiert. Die kurze Form, anwendbar für Längenangaben bis 127, besteht aus nur einem Oktett:

8	7	6	5	4	3	2	1
0	Length der folgenden Daten						

Die lange Form, die für Längenangaben über erforderlich ist, die die Zahl 127 übersteigen, besteht aus zwei oder mehr Oktetten, wobei das erste Oktett die Anzahl der nachfolgendenOktette angibt:

8	7	6	5	4	3	2	1
1	Anzahl der nun folgenden Length Oktette						

8	7	6	5	4	3	2	1
Höherwertigste Bits von **Length** der folgenden Daten zuerst							

Neben diesen sogenannten bestimmten (*definite*) Formen der Längenangabe, gibt es auch noch eine weniger gebräuchliche, unbestimmte (*indefinite*) Form: sie besteht aus einem Oktett, bei dem als einziges das Bit 8 gesetzt ist, und muß mit einem End-Of-Contents Tag (UNIVERSAL 0) abgeschlossen sein. Eine indefinite Form muss stets einen CONSTRUCTED Inhalt besitzen.

Der Daten-Wert (*Value*) der Datenstruktur, bestehend aus der Anzahl von Oktetten, wie vorgegeben durch die Längenangabe im Header, folgt nun auf den Header. Die Interpretation des Daten-Wertes hängt von *Class Form Tag* im Header ab.

Daten-Werte der *Form* 0 (PRIMITIVE), die mehrere Bytes zur Darstellung eines Elements erfordern, wie Integers oder Unicode, werden in ASN.1 grundsätzlich im *Big Endian* Format notiert, also mit dem höchstwertigen Byte voran, wie auch in Netzwerken üblich.

Bei *Form* 1 (CONSTRUCTED), beginnt der Daten-Wert mit einem neuerlichen Header. Bei dem Encoding von Daten werden noch Basic und Distinguished Encoding Rules unterschieden.

Bei den Distinguished Encoding Rules DER, darf es für eine Notations-Vorgabe nur genau eine Encoding-Darstellung geben. Diese ist, vereinfacht, stets das kürzeste mögliche Encoding. Daher *müssen* bei DER Tag-Nummern kleiner als 31 und Längenangaben bis 127 in jeweils genau einem Byte codiert sein. Weiterhin ist bei DER eine indefinite Längenangabe unzulässig. Analog gilt dieses auch für das Encoding der Daten-Werte, auf das hier wegen der Vielzahl von Basisdatentypen nicht näher eingegangen werden kann.

Bei den Basic Encoding Rules BER ist, vereinfacht ausgedrückt, jedes Encoding gültig, wenn es nur den gewünschten Wert liefert. Für cryptographische Anwendungen ist BER zumeist ungeeignet, da zu signierende Daten in vielfacher Form dargestellt werden könnten.

ASN.1-Datenstrukturen als APL Objekte

Der Header einer ASN.1-Datenstruktur besteht aus dem „Quartett“ *Class*, *Form*, *Tag* und *Length*.

Die drei ersten (*Class Form Tag*) erfordern eine Darstellung in einer expliziten APL Struktur, die hier als ASN.1-Identifier bezeichnet werden. (Die derzeitigen APL-Implementierungen können zu Variablen keine impliziten Anwender-Informationen vorhalten.) Da APL jedoch implizit die Längen-Information von Variablen verwaltet, ist es nicht unbedingt erforderlich, die *Length*-Information im ASN.1-Identifier noch einmal zu hinterlegen.

Daher sieht die naheliegende Darstellung der ASN.1-Datenstrukturen in APL so aus, dass Zwei- oder Mehr-Element-Vektoren gebildet werden, deren erstes Element der ASN.1-Identifier (*Class Form Tag*) ist. Die weiteren Elemente beinhalten dann die Daten-Werte.

Bei Objekten der *Form* PRIMITIVE, ist dieses genau ein (eingeschlossenes) Element, mit Ausnahme des NULLTAG, welches gemäß seiner Definition ausschließlich aus dem Identifier besteht.

Wenn die *Form* CONSTRUCTED ist, folgen auf den Identifier so viele weitere eingeschlossene Elemente, wie erforderlich sind, um die Bestandteile, etwa einer SET- oder SEQUENCE-Konstruktion aufzunehmen.

Um nicht in jeder Zuweisung einer ASN.1-Datenstruktur immer wieder (*Class*, *Form*, *Tag*) auf dem ersten Element angeben zu müssen, werden mit folgenden Zuweisungen ASN.1-Identifier global definiert:

⍝ Die gebräuchlichsten ASN.1-Identifier der Class UNIVERSAL:

```
BOOLEAN          ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_BOOLEAN
INTEGER          ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_INTEGER
BITSTRING        ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_BITSTRING
OCTETSTRING      ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_OCTETSTRING
NULLTAG         ←,⊂CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_NULLTAG
OID              ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_OID
UTF8STR          ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_UTF8STR
SEQUENCE         ← CLASS_UNIVERSAL FORM_CONSTRUCTED TAG_SEQUENCE
SET              ← CLASS_UNIVERSAL FORM_CONSTRUCTED TAG_SET
PRINTABLESTR     ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_PRINTABLESTR
T61STRING        ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_T61STRING
IA5STR           ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_IA5STR
UTCTIME          ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_UTCTIME
GENERALIZEDTIME← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_GENERALIZEDTIME
BMPSTR           ← CLASS_UNIVERSAL FORM_PRIMITIVE   TAG_BMPSTR
```

⍝ Eine Hilfsfunktion für Class `CONTEXT`, aus [n] wird `(CONTEXT n)`:

```
CONTEXT          ←{CLASS_CONTEXT   FORM_CONSTRUCTED ⍵}
```

⍝ Zusätzlich können wir für den Typ `BOOLEAN` noch gebrauchen:

```
FALSE            ← 0

TRUE             ← 1
```

Ein Encoding-Beispiel

Auf der Basis dieser Zuweisungen können wir nun, durch Ausfüllen aller Felder, eine Encoding-Instanz unseres anfänglichen Notations-Beispiels `Letter` erstellen. Zuerst werden die einzelnen Felder des Absenders zugewiesen. Jeder Identifier beschreibt Daten vom Typ UNIVERSAL 22 oder PRINTABLESTR:

```
⋄ ⋄ Name   ← PRINTABLESTR ∆Meiner Einer∇
⋄ ⋄ Street ← PRINTABLESTR ∆Meine Str. 123∇
⋄ ⋄ Town   ← PRINTABLESTR ∆Heimatstadt∇
```

Aus diesen drei Zuweisungen können wir nun die erste Sequenz konstruieren, die den Absender `Sender`, vom Typ UNIVERSAL 16 oder SEQUENCE beschreibt:

```
⋄ Sender   ← SEQUENCE Name Street Town
```

Hinweis:

Die beiden Diamonds ⋄ an jedem Zeilenanfang sollen zur besseren Lesbarkeit eine Einrückung erzwingen um, ähnlich wie bei den Kontroll-Strukturen, den logischen Aufbau der Zuweisungen zu visualisieren. Sie haben auf die syntaktische Interpretation keine Auswirkung.

Diesem folgt unter Verwendung der gleichen Identifier die Zuweisung des Empfängers **Addressee**:

```
⋄ ⋄ Name   ← PRINTABLESTR ∆Deiner Einer∇

⋄ ⋄ Street ← PRINTABLESTR ∆Deine Str. 999∇

⋄ ⋄ Town   ← PRINTABLESTR ∆Fremdstadt∇

⋄ Addressee ← SEQUENCE Name Street Town
```

Es bleiben noch die letzten beiden Felder von `Letter`. Der Identifier des **Date**-Feldes hat den Typ UNIVERSAL 23 oder UTCTIME und sein Inhalt wird im lokalen **ŒTS**-Format angegeben. Dieses Format kann man später mit der ASN.1 Codierungs-Funktion nach UTC transformieren :

```
⋄ Message ← T61STR ∆Grüße aus der Heimat∇
⋄ Date    ← UTCTIME (2002 1 18 17 30 0 0)
```

Nachdem nun alle Felder unseres Briefes vorliegen, können wir die genestete Hauptstruktur `Letter` zuweisen. Genauso wie `Sender` und **Addressee**, ist auch `Letter` eine SEQUENCE :

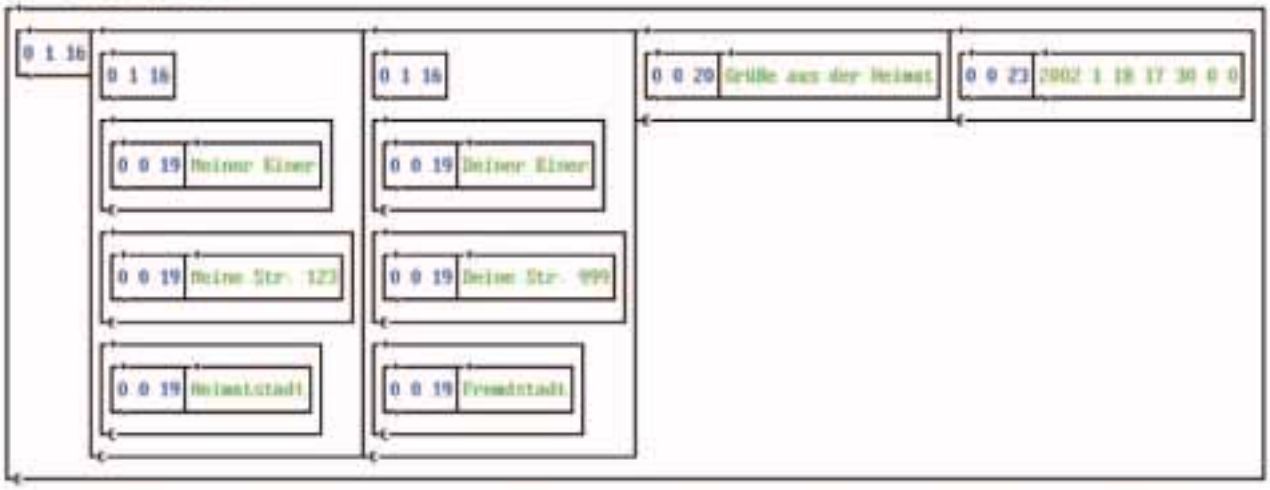
```
Letter     ← SEQUENCE Sender Addressee Message Date
```

Zur Kontrolle ist die genestete Struktur der Variablen `Letter`, ein 5-Elemente-Vektor der Tiefe 4, in der nachfolgenden Abbildung auf Seite 11 mit einer modifizierten Version der Dyalog APL Funktion **Display** dargestellt :

Hinweis:

Einige Nesting-Ebenen sind untereinander statt nebeneinander dargestellt, um auf diese Weise den Darstellungsplatz besser auszunutzen.

```
      ⍎Letter
┌→
      2 Display Letter
```



Nachdem die innersten Strukturen nach ASN.1 encodiert worden sind, sieht das vorliegende Resultat so aus:

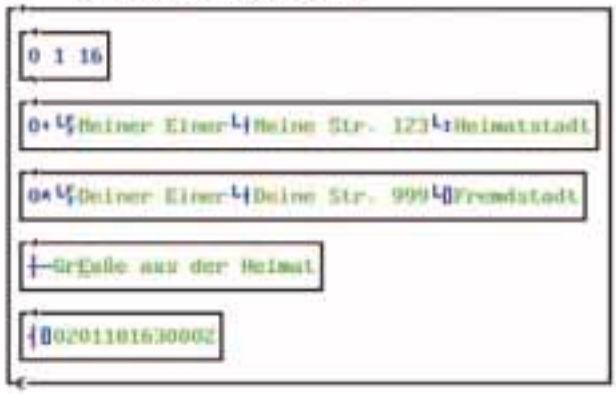
```
      ⍎3 Code Letter
┌→
      2 Display 3 Code Letter
```

Hinweis:

Das linke Argument der Funktion **Code** *legt die gewünschte Tiefe des Resultats fest.*

Im nächsten Schritt werden, durch Erzwingen der Tiefe 2, alle Elemente der Hauptsequenz in ASN.1-Vektoren encodiert:

```
      ⍎2 Code Letter
┌→
      1 Display 2 Code Letter
```



Hinweis:

Die nicht lesbaren Zeichen in den Vektoren sind die bereits encodierten ASN.1-Header, sowie nach T.61 gewandelte Umlaute.

Dieses ist nun der endgültige ASN.1-String als APL-Vektor. Er wird durch Vorgabe der Tiefe 1, der Tiefe von Zeichenvektoren erreicht :

Hinweis:

Die Darstellung ist auf die druckbare Breite der Seite begrenzt worden.

```
      ⍎1 Code Letter
1
      ⍕1 Code Letter
129
      ⍕⍕9 Display 1 Code Letter
┌──────────────────────────────────────────────────────────────────────────────┐
│0⍬0+⍬Meiner Einer⍬Meine Str. 123⍬Heimatstadt0*⍬Deiner Einer⍬Deine Str. 999⍬Fremds│
└──────────────────────────────────────────────────────────────────────────────┘
```

Der ASN.1-String im klassischen Dump-Format sieht wie folgt aus:

```
30 7F 30 2B 13 0C 4D 65 69 6E 65 72 20 45 69 6E   0.0+..Meiner Ein
65 72 13 0E 4D 65 69 6E 65 20 53 74 72 2E 20 31   er..Meine Str. 1
32 33 13 0B 48 65 69 6D 61 74 73 74 61 64 74 30   23..Heimatstadt0
2A 13 0C 44 65 69 6E 65 72 20 45 69 6E 65 72 13   *..Deiner Einer.
0E 44 65 69 6E 65 20 53 74 72 2E 20 39 39 39 13   .Deine Str. 999.
0A 46 72 65 6D 64 73 74 61 64 74 14 15 47 72 C8   .Fremdstadt..GrÈ
75 FB 65 20 61 75 73 20 64 65 72 20 48 65 69 6D   uûe aus der Heim
61 74 17 0D 30 32 30 31 31 38 31 36 33 30 30 30   at..020118163000
5A                                                Z
```

Hinweis:

Es ist lediglich ein Overhead von 22 Bytes notwendig, um alle Struktur-, Typ- und Längenangabe zu verwalten.

Die Funktion #.ASN1.Code

Im ersten Ansatz, ein ASN.1-Interface für APL zu entwickeln, hatte ich noch getrennte Funktionen `Encode` und `Encode` programmiert. Die `Decode`-Funktion hatte ein linkes Argument, mit dem die Tiefe angegeben wurde, in die constructed Tags aufzulösen waren.

Es soll hier nicht unerwähnt bleiben, dass diese Funktionen oftmals nicht die notwendige Performance erreichten, um Anwender-Erfordernisse zufriedenzustellen. Dieses betraf insbesondere Situationen, in denen umfangreiche ASN.1-Sequenzen, bisweilen Hunderte von Kilobytes groß, gefüllt mit ganzen Adressbüchern von Zertifikaten aufzulösen waren. (Das Decodieren von ASN.1-Headern, rekursiv Bit für Bit, ist für array-orientierte Interpreter-Sprachen nun einmal keine sonderlich geeignete Aufgabe.)

Aus diesem Grund habe ich diese Funktionen in der Sprache Assembler, unter Verwendung des `⎕NA`-Interfaces von Dyalog APL, als 32-Bit-DLL neu geschrieben. Die Übergabe-Argumente der DLL werden als genestete APL-Datentypen ausgetauscht. Die Funktionen sind somit im Stande, APL-Strukturen in Assembler-Geschwindigkeit zu verstehen und auch selbst aufzubauen.

Bei dieser Neuentwicklung ist zugleich ein Redesign erfolgt, bei dem die beiden ehemaligen Funktionen durch eine einzige Funktion `Code` ersetzt worden sind.

Unabhängig davon, in welcher Form das übergebene Argument vorliegt, ob als genestete APL-Struktur, oder als, auch partiell schon decodierter, ASN.1-String, das Resultat von `Code` wird stets auf die Tiefe (-) codiert zurückgegeben, die mit dem linken Argument spezifiziert worden ist, soweit natürlich die übergebene ASN.1-Struktur diese Tiefe überhaupt besitzt.

Das bedeutet, dass der Ausdruck `1 Code ARG` stets ein vollständiges Encodieren nach ASN.1 als Vektor zurückliefert. `9999 Code ARG` gibt stets eine vollständige Decodierung in eine genestete APL-Struktur zurück, unabhängig vom aktuell übergebenen Zustand von `ARG`. Da ein Ergebnis der Tiefe `0` (simpler Skalar) semantisch nicht möglich ist, darf die `0` statt der `9999` zur Decodierung in die maximale Tiefe als linkes Argu-ment übergegeben werden. Ein vollständiges Decodieren nach APL nested wird daher immer mit `0 Code ARG` erzielt. Dieses ist auch der Default für das linke Argument, wenn `Code` monadisch ausgeführt wird.

Wird das linke Argument mit negativem Vorzeichen angegeben, versucht `Code` keine spekulativen Decodierungen in Oktettstrings und in byte-bündigen Bitstrings.

Optional kann ein zweites linkes Argument übergeben werden, das Optionen beim Codieren von *Tags* der *Class* UNIVERSAL setzt. Die vollständige Syntax der Funktion `Code` sieht daher wie folgt aus:

```
RES←Depth UnivTagOptions Code ARG
```

Hinweis:

Mancher wundert sich, wozu es Sinn macht, etwa ein Zertifikat mit **Cert„2 Code Cert** nur partiell zu decodieren.

Der Grund ist, dass einige cryptographische Anwendungen die Bildung von Hash-Werten (Digests) über derartige partiell decodierte Teilsequenzen erfordern.

Ohne den obigen Ausdruck etwa könnte man die Signatur eines Zertifikats nicht bilden oder überprüfen.

Denn hierzu muss der gepaddete Hash-Wert der „toBeSignedCertificate"-Sequenz mit der durch den öffentlichen Schlüssel des Aussteller-Zertifikats decrypteten „signature" verglichen werden.

Object Identifier

Darstellung einiger Object Identifier mit ihrer Hierarchie in der ASN.1-Syntax-Notation:

OID	::= OBJECT IDENTIFIER	
ccitt	OID ::= {0}	***– Einstiegspunkt***
ccitt-identified-organization	OID ::= {0 4}	
iso	OID ::= {1}	***– Einstiegspunkt***
iso-standard	OID ::= {1 0}	
iso-member-body	OID ::= {1 2}	
iso-member-body-de	OID ::= {1 2 276}	*– Zuständig DIN*
iso-member-body-us	OID ::= {1 2 840}	*– Zuständig NIST*
rsadsi	OID ::= {1 2 840 113549}	*– Start RSA*
pkcs	OID ::= {rsadsi 1}	
pkcs-1	OID ::= {pkcs 1}	*– Aus der PKCS#1*
rsaEncryption	OID ::= {pkcs-1 1}	
md2WithRSAEncryption	OID ::= {pkcs-1 2}	
md5WithRSAEncryption	OID ::= {pkcs-1 4}	
sha1WithRSAEncryption	OID ::= {pkcs-1 5}	
pkcs-7	OID ::= {pkcs 7}	*– Aus der PKCS#7*
data	OID ::= {pkcs-7 1}	
signedData	OID ::= {pkcs-7 2}	
digestedData	OID ::= {pkcs-7 5}	
encryptedData	OID ::= {pkcs-7 6}	
pkcs-9-at-emailAddress	OID ::= {pkcs 9 1}	*– Aus der PKCS#9*
iso-identified-organization	OID ::= {1 3}	
iso-org-dod-internet-private-enterprises	OID ::= {1 3 6 1 4 1}	*– Liste der IANA*
microsoft	OID ::= {1 3 6 1 4 1 311}	*– Start Microsoft*
microsoft-spcStatementType	OID ::= {microsoft 2 1 11}	
microsoft- spcSpOpusInfo	OID::= {microsoft 2 1 12}	
datev	OID ::= {1 3 6 1 4 1 3744}	*– Start Datev*
joint-iso-ccitt	OID ::= {2}	***– Einstiegspunkt***
ds	OID ::= {2 5}	
attributeType	OID ::= {ds 4}	*– Aus der X.520*
id-at	OID ::= attributeType	
id-at-commonName	OID ::= {id-at 3}	
id-at-countryName	OID ::= {id-at 6}	
id-at-localityName	OID ::= {id-at 7}	
id-at-stateOrProvinceName	OID ::= {id-at 8}	
id-at-streetAddress	OID ::= {id-at 9}	
id-at-organizationName	OID ::= {id-at 10}	
id-at-organizationalUnitName	OID ::= {id-at 11}	
certificateExtension	OID ::= {ds 29}	*– Aus der X.509*
id-ce	OID ::= certificateExtension	
id-ce-basicConstraints	OID ::= {id-ce 19}	

Die *Object Identifier* gehören, insbesondere wegen des häufigen Vorkommens, zu den bedeutendsten Elementen in ASN.1. Wie bereits erwähnt, sind sie von der ITU als Type UNIVERSAL 6, OBJECT IDENTIFIER definiert, oder kurz OID.

OIDs werden dazu verwendet, die Bedeutung eines nachfolgenden Informations-Objektes festzulegen, oder den Zustand einer Information anzuzeigen. Um dieses leisten zu können, müssen OIDs eindeutig sein. Die CCITT (die heutige ITU-T) und die ISO, sowie deren Mitglieder haben mittlerweile einige tausend OIDs definiert. Wie die hier exemplarisch aufgeführten, können OIDs verschiedenstes beschreiben: Datenklassen, Institutionen, Definitionen, etc.

Ein Object Identifier, dessen Bedeutung durch eine, bisweilen selbst in ASN.1 abgefasste Beschreibung, genau vorgegeben ist, erklärt den Zweck des zugehörigen Daten-Objekts und macht es damit eindeutig interpretierbar.

Zum Identifizieren unbekannter OIDs, oder um eigene Entries aufzubauen, sind Harald Tveit Alvestrands offene Sammlung [http://www.alvestrand.no/objectid/top.html] und die Liste der Enterprise-Numbers bei der IANA [http://www.iana.org/] hilfreich.

Der Daten-Wert eines OIDs besteht aus einer Abfolge von nicht negativen Zahlen, und lässt sich in APL zweckmäßigerweise als numerischer Vektor darstellen. Die Sequenz dieser Zahlen ist, ausgehend von den drei einstelligen Einstiegspunkten, CCITT(0), ISO(1) und deren gemeinsamen Normen(2), hierarchisch organisiert.

Um das `Letter`-Beispiel auf OIDs umzustellen, könnten wir die Inhalte von Address jeweils durch folgende SET OF SEQUENCE Abfolgen ersetzen:

```
OID id-at-commonName    und  Name,
OID id-at-streetAddress und  Street,
OID id-at-localityName  und  Town.
```

⍝ Mögliche Zuweisung einiger dieser OIDs zur Verwendung in APL:

```
pkcs_1_rsaEncryption            ← 1 2 840 113549 1 1 1

pkcs_1_md2withRSAEncryption     ← 1 2 840 113549 1 1 2
pkcs_1_md5withRSAEncryption     ← 1 2 840 113549 1 1 4
pkcs_1_sha1withRSAEncryption    ← 1 2 840 113549 1 1 5
pkcs_7_data                     ← 1 2 840 113549 1 7 1
pkcs_7_signedData               ← 1 2 840 113549 1 7 2
pkcs_7_digestedData             ← 1 2 840 113549 1 7 5
pkcs_7_encryptedData            ← 1 2 840 113549 1 7 6
pkcs_9_at_emailAddress          ← 1 2 840 113549 1 9 1
ms_spcStatementType             ← 1 3 6 1 4 1 311 2 1 11
ms_spcSpOpusInfo                ← 1 3 6 1 4 1 311 2 1 12
id_at_commonName                ← 2 5 4 3
id_at_countryName               ← 2 5 4 6
id_at_localityName              ← 2 5 4 7
id_at_stateOrProvinceName       ← 2 5 4 8
id_at_streetAddress             ← 2 5 4 9
id_at_organizationName          ← 2 5 4 10
id_at_organizationalUnitName    ← 2 5 4 11
id_ce_basicConstraints          ← 2 5 29 19
```

Object-Type Name

Ein X.509-Zertifikat enthält zwei Adressen. Die erste ist die Adresse des Ausstellers, der den Inhalt zertifiziert, und die zweite ist die Adresse der Person, über die das Zertifikat ausgestellt ist.

Der Aufbau der Adressen ist in ITU X.501 und X.520 als Object-Type Name definiert:

```
Name ::=                    CHOICE {
    rdnSequence                     SEQUENCE OF {
        relativeDistinguishedName   SET OF {
                    attributeTypeAndValue SEQUENCE {
                        type                    OID,
                        value                   @type }}}}
```

Hierzu noch ein paar Erläuterungen. Die Formulierung *@type* zeigt an, dass der Typ von *value* in der Beschreibung zum OID *type* festgelegt ist, oder anders formuliert, ANY DEFINED BY *type*.

Für die typischen Textfelder einer Adresse, wie CommonName oder LocalityName ist in der X.520 *DirectoryString* angegeben, der als Auswahl (CHOICE) aus *PrintableString*, *T61String*, *UTF8String*, *BMPString* oder *UniversalString* definiert ist, wobei der einfachste Objekt-Typ, der alle zur Darstellung des Textfeldes nötigen Zeichen in seinem Zeichensatz-Repertoire hat, zu bevorzugen ist. Allerdings wird mittlerweile von T61String wegen nicht einheitlich implementierter Encodings abgegangen.

Weshalb die Festlegung zu Name so unsäglich verschachtelt ausfallen musste, wird wohl für immer ein Geheimnis der ITU bleiben. Der beabsichtigte Zweck, beliebige Adressfelder anzuordnen, hätte jedenfalls ohne CHOICE, die weder eine Auswahl anbietet, noch sich auf das Encoding auswirkt, und auch ohne SET OF erheblich einfacher erreicht werden können.

In Anlehnung an das **Letter**-Beispiel sieht ein Adressblock mit den dortigen Feldern dann so aus:

```
Name ::=                    SEQUENCE {
        relativeDistinguishedName   SET {
                attributeTypeAndValue       SEQUENCE {
                        id-at-commonName            OID,
                        name                        PRINTABLESTR }}

        relativeDistinguishedName   SET {
                attributeTypeAndValue       SEQUENCE {
                        id-at-streetAddress         OID,
                        street                      PRINTABLESTR }}

        relativeDistinguishedName   SET {
                attributeTypeAndValue       SEQUENCE {
                        id-at-localityName          OID,
                        town                        PRINTABLESTR }}
```

Die Struktur eines signierten X.509-Zertifikats

Ein signiertes X.509-Zertifikat besteht aus einer Sequenz von drei Elementen: Dem zu signierenden Zertifikat, der Angabe, mit welchem Algorithmus es signiert ist, und der Signatur.

Diese ist als Bitstring-Primitive definiert, die beiden anderen sind Constructed-Typen:

```
Certificate                 ::= SEQUENCE {
        toBeSignedCertificate       ToBeSignedCertificate,      -- „Unterschriftsvorlage“
        algorithmIdentifier         AlgorithmIdentifier,        -- Unterschrieben mit ...
        signature                   BITSTRING }                 -- Unterschrift
```

Der Object-Type *ToBeSignedCertificate* wird auch als *TBSCertificate* oder als *CertificateInfo* bezeichnet und beherbergt sämtliche Inhalte des Zertifikats. Er wird wie nachfolgend dargestellt definiert:

```
ToBeSignedCertificate       ::= SEQUENCE {
        version                     [0] Version DEFAULT v1,     -- Erst seit Version v2
        serialNumber                INTEGER,                    -- Seriennummer
        signature                   AlgorithmIdentifier,        -- Zu unterschreiben mit ...
        issuer                      Name,                       -- Ausgestellt von ...
        validity                    Validity,                   -- Gültigkeits-Zeitraum
        subject                     Name,                       -- Ausgestellt für ...
        subjectPublicKeyInfo        SubjectPublicKeyInfo,       -- Öffentlicher Schlüssel
        issuerUniqueID              [1] BITSTRING OPTIONAL,     -- Erst ab Version v2
        subjectUniqueID             [2] BITSTRING OPTIONAL,     -- Erst ab Version v2
        extensions                  [3] Extensions OPTIONAL },  -- Erst ab Version v3
```

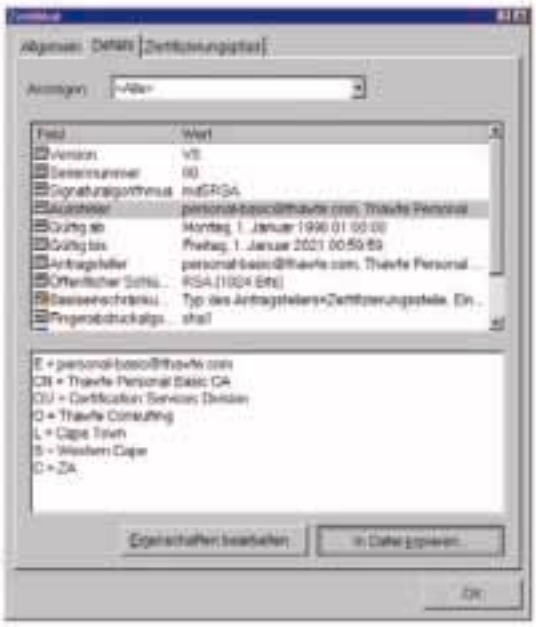
Die vorkommenden Objekte *Version*, *AlgorithmIdentifier*, *Validity*, *SubjectPublicKeyInfo*, und *Extensions* sind in der nachstehenden Auflistung teilweise vereinfacht dargestellt undwerden separat definiert:

```
Version             ::= INTEGER { v1(0), v2(1), v3(2) }
Validity            ::= SEQUENCE {
        notBefore           CHOICE { UTCTIME,  GENERALIZEDTIME },
        notAfter            CHOICE { UTCTIME,  GENERALIZEDTIME }}
```

```
SubjectPublicKeyInfo        ::= SEQUENCE {
        algorithm                   AlgorithmIdentifier,
        subjectPublicKey            BITSTRING }
AlgorithmIdentifier         ::= SEQUENCE {
        algorithm                   OID,
        parameters                  @algorithm OPTIONAL }
Extensions                  ::= SEQUENCE OF Extension
Extension                   ::= SEQUENCE {
        extnID                      OID,
        critical                    BOOLEAN DEFAULT FALSE,
        extnValue                   OCTETSTRING }
```

Encoding eines signierten X.509-Zertifikats

Als Beispielvorlage zum Encodieren eines X.509-Zertifikats dient hier das Thawte Personal Basic CA (siehe untenstehende Abbildung), das auf Windows-Rechnern in der Zertifikatsverwaltung unter den vertrauenswürdigen Stammzertifizierungsstellen vorhanden ist.



Dieses kann entweder mit dem Button unter CAThawte.CER in eine Datei kopiert, und anschließend als APL-Vektor geladen werden:

```
CAThawte←#.Win.File.Load∇CAThawte.CER∇
```

oder mit dem APL-Code auf der nächsten Seite selbst aufgebaut werden.

Da dieses selbst aufgebaute Zertifikat auch eine Signatur (signature) benötigt, müssen wir zuerst ein RSA-Schlüsselpaar generieren, mit dem wir es unterschreiben können. Aus diesem Schlüsselpaar können wir auch den öffentlichen Schlüssel (subjectPublicKey) entnehmen. Abgesehen von PublicKey und Signatur, ist das selbsterstellte mit dem Original völlig identisch. Es handelt sich hier um ein v3 Zertifikat, das in ASN.1-Syntax-Notation, schematisch vereinfacht, etwa wie folgt aussieht:

```
CAThawte     ::=     SEQUENCE {
        tBSCertificate          SEQUENCE {
                version                     [0] INTEGER v3(2),
                serialNumber                INTEGER (0),
                signature                   SEQUENCE {OID (md5WithRSAEncryption), NULL},
                issuer                      Name,
                validity                    SEQUENCE {
                        notBefore                   UTCTIME (1996 01 01 01 00 00 000),
                        notAfter                    UTCTIME (2021 01 01 00 59 59 000) },
                subject                     Name (issuer),
                SubjectPublicKeyInfo        SEQUENCE {
                        algorithm                   SEQUENCE {OID (rsaEncryption), NULL},
                        subjectPublicKey            BITSTRING }
                 extensions                 [3] SEQUENCE OF { SEQUENCE {
                        extnID                      OID (id-ce-basicConstraints),
                        critical                    BOOLEAN (TRUE),
                        extnValue                   OCTETSTRING}}},
        algorithmIdentifier     SEQUENCE {OID (md5WithRSAEncryption), NULL},
        signature               BITSTRING }
```

⍝ Für SubjectPublicKey und zum Signieren benötigen wir ein Schlüsselpaar:

```
PrivatKey←#.ASN1.PKCS1.KeyEncode 8↑KeyPair←#.ASN1.PKCS1.KeyGenerate 1024
PublicKey←#.ASN1.PKCS1.KeyEncode 2↑KeyPair
```

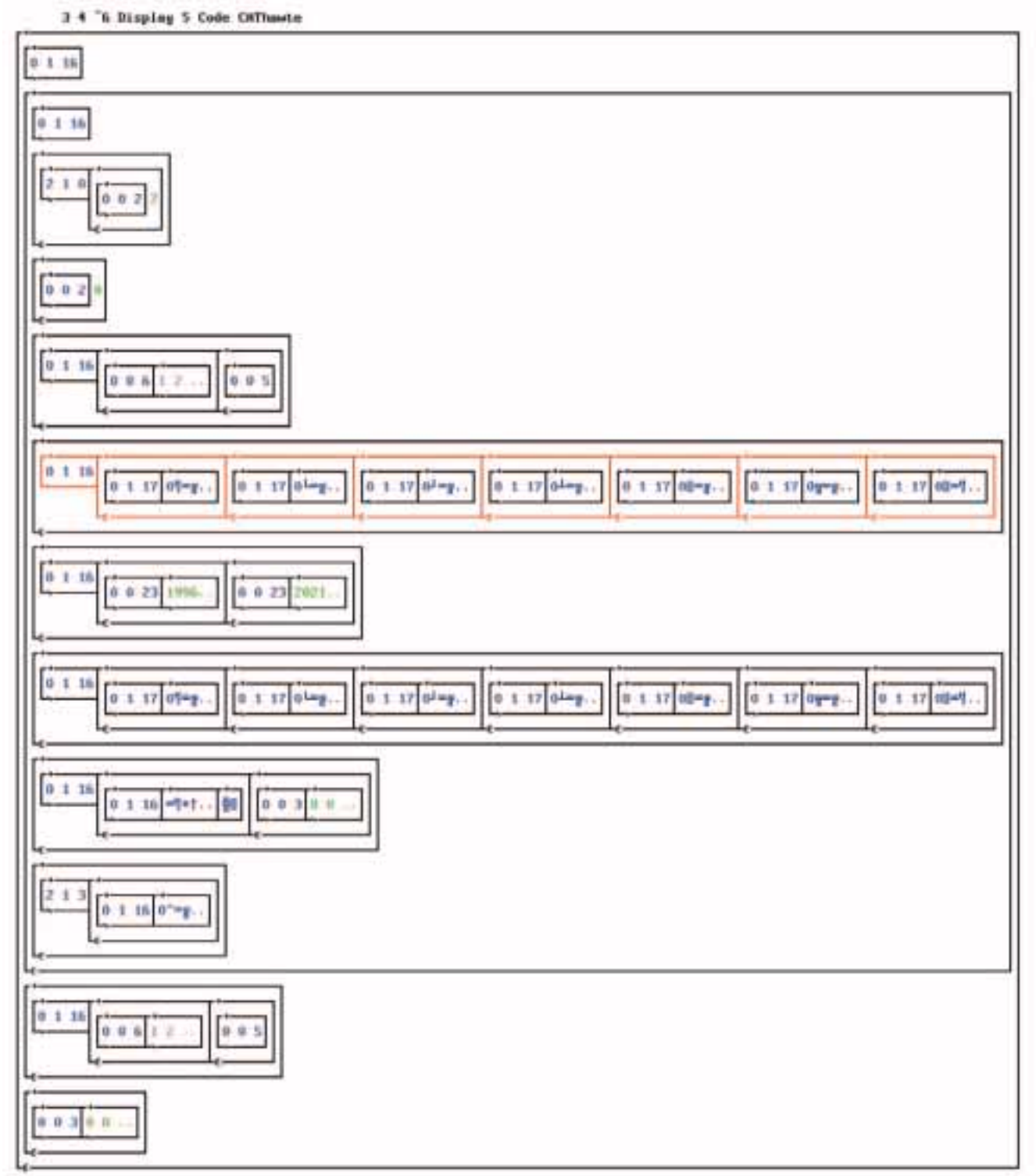
⍝ Nun können wir das zu unterzeichnende TBSCertificate zusammenbauen:

```
⋄ ⋄ ⋄ v1 v2 v3←0 1 2
⋄ ⋄ Version←(CONTEXT 0)(INTEGER v3)
⋄ ⋄ SerialNumber←INTEGER 0
⋄ ⋄ ⋄ Algorithm←OID pkcs_1_md5withRSAEncryption
⋄ ⋄ ⋄ Parameters←NULLTAG
⋄ ⋄ Signature←SEQUENCE Algorithm Parameters
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_countryName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕ZA⍕
⋄ ⋄ ⋄ CountryName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_stateOrProvinceName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕Western Cape⍕
⋄ ⋄ ⋄ StateOrProvinceName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_localityName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕Cape Town⍕
⋄ ⋄ ⋄ LocalityName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_organizationName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕Thawte Consulting⍕
⋄ ⋄ ⋄ OrganizationName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_organizationalUnitName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕Certification Services Division⍕
⋄ ⋄ ⋄ OrganizationalUnitName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID id_at_commonName
⋄ ⋄ ⋄ ⋄ Value←PRINTABLESTR⍕Thawte Personal Basic CA⍕
⋄ ⋄ ⋄ CommonName←SET(SEQUENCE AttributeType Value)
⋄ ⋄ ⋄ ⋄ AttributeType←OID pkcs_9_at_emailAddress
⋄ ⋄ ⋄ ⋄ Value←IA5STR⍕personal-basic@thawte.com⍕
⋄ ⋄ ⋄ EmailAddress←SET(SEQUENCE AttributeType Value)
⋄ ⋄ Issuer←SEQUENCE CountryName StateOrProvinceName LocalityName ..

.. OrganizationName OrganizationalUnitName CommonName EmailAddress
⋄ ⋄ ⋄ NotBefore←UTCTIME(1996 1 1 1 0 0 0)
⋄ ⋄ ⋄ NotAfter←UTCTIME(2021 1 1 0 59 59 0)

⋄ ⋄ Validity←SEQUENCE NotBefore NotAfter
⋄ ⋄ Subject←Issuer ⍝ Bei Rootzertifikaten sind Subject und Issuer identisch
⋄ ⋄ ⋄ ⋄ Algorithm←OID pkcs_1_rsaEncryption
⋄ ⋄ ⋄ ⋄ Parameters←NULLTAG
⋄ ⋄ ⋄ Algorithm←SEQUENCE Algorithm Parameters
⋄ ⋄ ⋄ SubjectPublicKey←BITSTRING PublicKey
⋄ ⋄ SubjectPublicKeyInfo←SEQUENCE Algorithm SubjectPublicKey
⋄ ⋄ ⋄ ⋄ ExtnId←OID id_ce_basicConstraints
⋄ ⋄ ⋄ ⋄ Critical←BOOLEAN TRUE
⋄ ⋄ ⋄ ⋄ ExtnValue←OCTETSTRING(SEQUENCE(BOOLEAN TRUE))
⋄ ⋄ ⋄ BasicConstraints←SEQUENCE ExtnId Critical ExtnValue
⋄ ⋄ Extensions←(CONTEXT 3)(SEQUENCE BasicConstraints)
⋄ TBSCertificate←1 #.ASN1.Code SEQUENCE Version SerialNumber Signature ..
.. Issuer Validity Subject SubjectPublicKeyInfo Extensions
```

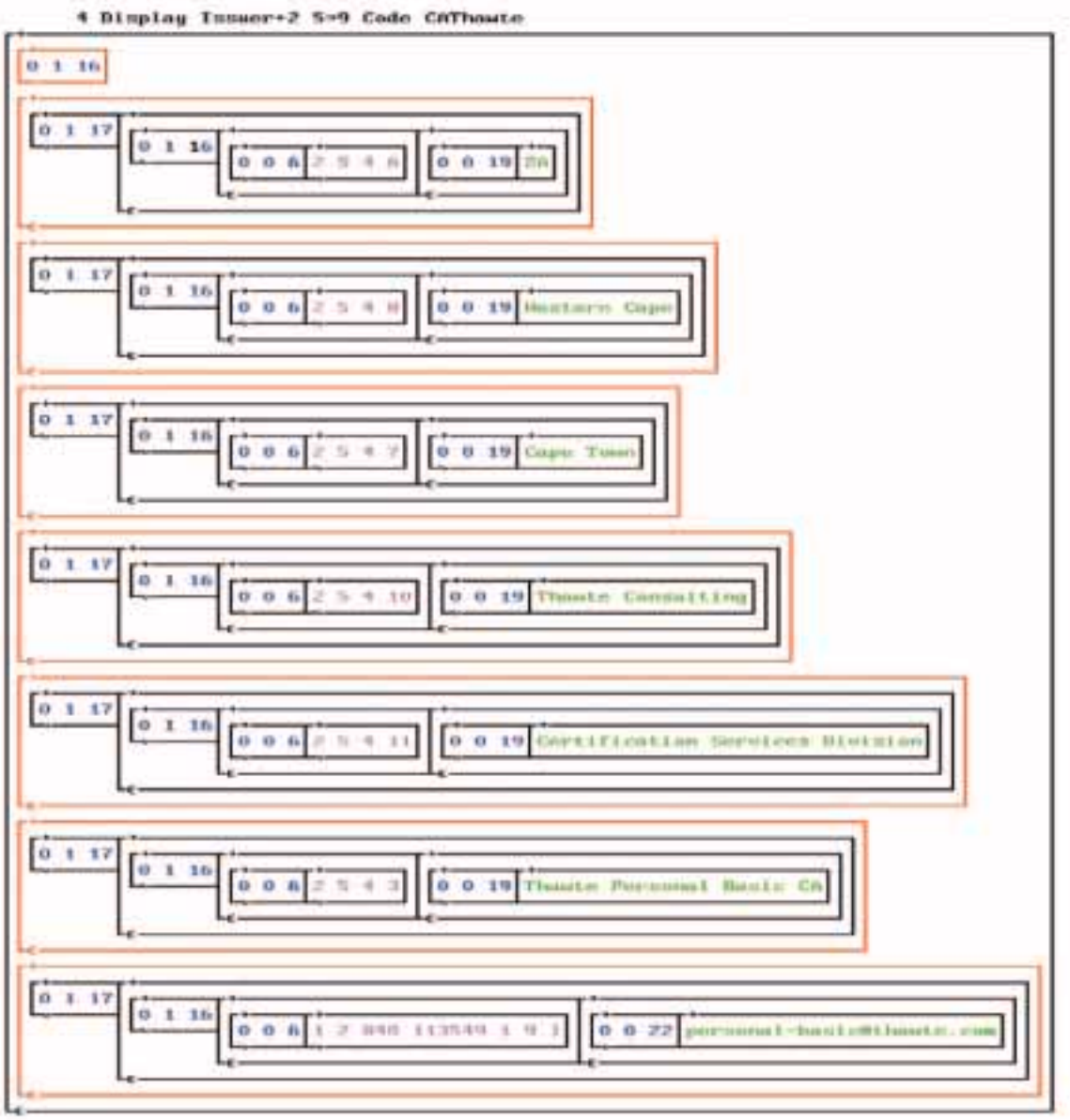
⍝ Noch die Signatur darunter und das X.509-konforme Zertifikat CAThawte ist fertig:

```
⋄ ⋄ Algorithm←OID pkcs_1_md5withRSAEncryption
⋄ ⋄ Parameters←NULLTAG
⋄ AlgorithmIdentifier←SEQUENCE Algorithm Parameters
⋄ Signature←BITSTRING(TBSCertificate #.Crypt.Sign PrivatKey #.Win.CALG_MD5)
CAThawte←1 #.ASN1.Code SEQUENCE TBSCertificate AlgorithmIdentifier Signature
```

Der schematische Aufbau der *Certificate*-SEQUENCE (0 1 16) mit den, verkürzt dargestellten, drei Elementen darunter, *TBSCertificate*, *AlgorithmIdentifier* und *Signature,* kann beim Decodieren bis auf die Tiefe 5 noch übersichtlich dargestellt werden:



Mit der Pick-Funktion (⊃) wird aus dem vollständig decodierten Zertifikat der *Issuer* als fünftes vom zweiten Element, (die Indices sind wegen der Identifier um Eins erhöht), selektiert (siehe folgende Abbildung auf Seite 19):



Durch ein weiteres Pick, auf den *Issuer* angewandt, erhalten wir den Aussteller-Namensblock, den die Zertifikatsverwaltung anzeigt, als eine Matrix von OIDs und Texten (siehe nachstehende Auflistung):

```
((1↓⍳⍴Issuer)∘.,(2 2 2)(2 3 2))⊃¨⊂Issuer
 2 5 4 6                ZA
 2 5 4 8                Western Cape
 2 5 4 7                Cape Town
 2 5 4 10               Thawte Consulting
 2 5 4 11               Certification Services Division
 2 5 4 3                Thawte Personal Basic CA
 1 2 840 113549 1 9 1   personal-basic@thawte.com
```

Ein APL-Arbeitsbereich in ASN.1-Notation

Bisher wurde gezeigt, wie ASN.1-Strukturen in APL dargestellt werden können.

Die nachstehende Darstellung enthält einen Vorschlag für den umgekehrten Ansatz, wie ein ganzer APL-Arbeitsbereich vollständig in ASN.1-Notation ausgedrückt, und dementsprechend auch portiert werden kann:

```
AplFramework
DEFINITIONS ::=
BEGIN

AplWorkspace ::=    SEQUENCE {
        id-apl-ws               OID,
        dateOfSaving            CHOICE { UTCTIME,  GENERALIZEDTIME },
        data                    SEQUENCE {
                aplWsName               CHARACTERSTRING,
                aplObjects              AplObjects }}

AplObjects   ::=    SET OF {
        aplObject               SEQUENCE {
                objectName              CHARACTERSTRING,
                objectValue             ObjectValue }}

ObjectValue  ::=    CHOICE {
        variable                [2] SEQUENCE {
                variableRank            SEQUENCE OF INTEGER,
                variableData            CHOICE {
                        variableDataAlpha     CHARACTERSTRING,
                        variableDataBinary    BITSTRING,
                        variableDataInteger   SEQUENCE OF INTEGER,
                        variableDataReal      SEQUENCE OF REAL,
                        variableNested        SEQUENCE OF ObjectValue }},
        function                [3] SEQUENCE OF CHARACTERSTRING,
        nameSpace               [9] AplObjects }

CHARACTERSTRING ::= CHOICE {
        printableString         PRINTABLESTRING,
        teletexString           T61STR,
        bmpString               BMPSTRING }

-- Die APL Zeichen sind in Unicode Standard (BMPSTRING) im Bereich 2280-23FF definiert --

iso-org-dod-internet-private-enterprises-apl OID ::= {1 3 6 1 4 1 9999} -- Bei LANA beantragen!
id-apl                                       OID :: = iso-org-dod-internet-private-enterprises-apl
id-apl-ws                                    OID :: = { id-apl 1 }

END
```

Quellenangaben

ITU-T: X.200 (07/94) The Basic Reference Model (OSI)
ITU-T: X.208 (12/88) Specification of ASN.1
ITU-T: X.209 (12/88) Specification of Basic Encoding Rules for ASN.1
ITU-T: X.501 (11/93) The Directory: Models. UsefulDefinitions InformationFramework SchemaAdministration BasicAccessControl DSAOperationalAttributeTypes OperationalBindingManagement EnhancedSecurity
ITU-T: X.509 (03/00) The Directory: Public-Key and Attribute Certificate Frameworks. AuthenticationFramework CertificateExtensions AttributeCertificate-Definitions AlgorithmObjectIdentifiers
ITU-T: X.511 (11/93) The Directory: Abstract Service Definition. DirectoryAbstractService
ITU-T: X.518 (11/93) The Directory: Procedures for Distributed Operation. DistributedOperations HierarchicalOperationalBindings
ITU-T: X.519 (11/93) The Directory: Protocol Specifications. DirectoryAccessProtocol DirectorySystemProtocol DirectoryInformationShadowProtocol DirectoryOperationalBindingManagementProtocol ProtocolObjectIdentifiers DirectoryOperationalBindingTypes DirectorySecurityExchanges
ITU-T: X.520 (11/93) The Directory: Selected Attribute Types. SelectedAttributeTypes UpperBounds
ITU-T: X.680 (07/94) ASN.1 Specification of Basic Notation. ASN1-CHARACTER-MODULE
ITU-T: X.690 (07/94) ASN.1 Spec. of Basic/Canonical/Distinguished Encoding Rules
RSA: PKCS #1 (09/99) RSA Cryptography Standard (v2.1)
RSA: PKCS #7 (05/97) Cryptographic Message Syntax Standard (v1.6)
RSA PKCS #9 (11/93) Selected Attribute Types (v1.1)

Überarbeitete deutsche Übersetzung des Vortrags vom APL Berlin 2000 Kongress.

Kontakt

Peter-Michael Hager, HAGER-ELECTRONICS GmbH, Hamburger Str. 97, 44135 Dortmund, eMail: hager@dortmund.net

ANHANG

Die Funktionen im Namespace #.ASN1

Funktion #.ASN1.Init

Init sollte vor der Verwendung von Variablen oder weiteren Funktionen aus diesem Namespace aufgerufen werden, um die Anbindung der DYASN1.DLL zu ermöglichen, und um die folgenden Variablen (vergleiche hierzu die nachstehende Auflistung) in `#.ASN1` zu etablieren, sofern sie noch nicht vorhanden sind:

```
BITSTRING          PRINTABLESTR       TAG_EMBEDDED_PDV   TAG_UTCTIME
BMPSTR             REAL               TAG_ENUMERATED     TAG_UTF8STR
BOOLEAN            SEQUENCE           TAG_EOC
TAG_VIDEOTEXSTR
CONTEXT            SET                TAG_EXTERNAL       TAG_VISIBLESTR
EMBEDDED_PDV       T61STR             TAG_GENERALIZEDTIME UTO_ANSI
ENUMERATED         TRUE               TAG_GENERALSTR     UTO_AUTO
EOC                UNIVERSALSTR       TAG_GRAPHICSTR     UTO_FMT
EXTERNAL           UTCTIME            TAG_IA5STR         UTO_HEX
FALSE              UTF8STR            TAG_INTEGER        UTO_I32
GENERALIZEDTIME    VIDEOTEXSTR        TAG_NULLTAG        UTO_I48
GENERALSTR         VISIBLESTR         TAG_NUMERICSTR     UTO_I53
GRAPHICSTR         CLASS_APPLICATION  TAG_OBJDESCRIPTOR  UTO_LOCAL
IA5STR             CLASS_CONTEXT      TAG_OCTETSTRING    UTO_NUM
INTEGER            CLASS_PRIVATE      TAG_OID            UTO_SPCALL
NULLTAG            CLASS_UNIVERSAL    TAG_PRINTABLESTR   UTO_SPCSEQ
NUMERICSTR         FORM_CONSTRUCTED   TAG_REAL           UTO_STR
OBJDESCRIPTOR      FORM_PRIMITIVE     TAG_SEQUENCE       UTO_WIDE
OCTETSTRING        TAG_BITSTRING      TAG_SET            UTO_ZULU
OID                TAG_BMPSTR         TAG_T61STR         UnivTagOptions
OidTab             TAG_BOOLEAN        TAG_UNIVERSALSTR
```

Funktion #.ASN1.InitOidTab

InitOidTab baut die Matrix **OidTab** neu auf. Dieses wird bei Nachträgen in dieser Funktion notwendig. **OidTab** besteht aus vier Spalten von Vektoren mit der OID-Nummer, dem OID-String, sowie einer englischen und deutschen Bezeichnung.

Funktion #.ASN1.Exit

Exit sollte nach der letzten Verwendung des Namespace zur Bereinigung aufgerufen werden, um die DLL-Anbindungen und Variablen, ausgenommen **OidTab**, zu löschen. Dieses empfiehlt sich besonders vor dem Sichern eines Arbeitsbereichs.

Die Funktionen #.ASN1.LetterSample und #.ASN1.CAThawteSample

Diese Funktionen enthalten die Beispiele aus den vorangegangenen Kapiteln und können als Programmiervorlagen dienen. In produktiver Umgebung werden sie nicht benötigt.

Funktion #.ASN1.Adjust

Adjust ist eine monadische Funktion, die einen ASN.1-String in Form eines Character-Vektors für Argument und Ergebnis hat. Sie entfernt aus dem Argument überschüssige Zeichen, die sich an eine ASN.1-Sequenz anschließen und gibt den String längenkorrigiert zurück.

Diese Funktion ist bei Strings die von cluster-formatierten Smart Cards gelesen werden notwendig, um den String von der Cluster-Größe auf die ASN.1 String-Länge zu kürzen.

Bei ASN.1-Strings, die über möglicherweise geteilte TCP-Pakete empfangen werden, kann **Adjust** darüber Auskunft geben, ob der Empfang weiterer Pakete abzuwarten ist, und wie viele Bytes beim Empfang aneinandergereihter Sequenzen zu jeder einzelnen gehören.

Bei ungültigen Argumenten oder unvollständigen Strings ist das Resultat ein leerer Vektor und **#.RCode** ist gesetzt.

Funktion #.ASN1.Code

Code ist eine diadische Funktion mit optionalem linken Argument. So, wie das rechtsseitige Argument **ARG**, ist auch das Ergebnis **RES** eine ASN.1-Struktur in beliebiger Darstellung, vom Character-Vektor, bis zur genesteten APL-Struktur. Zusätzlich darf das Argument **ARG** auch noch als boolescher Vektor vorliegen.

Das linke Argument kann ein Vektor mit bis zu zwei Elementen entsprechend folgender Syntax sein:

```
RES←Code ARG
RES←Depth Code ARG
RES←Depth UnivTagOptions Code ARG
```

Der Betrag des ersten Elements **Depth** gibt die maximale die Tiefe (-) des Resultats an, wobei der Default-Wert 0 einer beliebigen Tiefe entspricht. Bei negativem **Depth** ist spekulatives Decodieren disabled.

Mit dem zweiten Element **UnivTagOptions** können Optionen zum Decodieren und Encodieren von Tags aus der Class Universal gesetzt werden. **UnivTagOptions** ist ein bis zu 30 Elemente langer Vektor numerischer oder leerer Elemente.

Jedes Element ist mit einem Universal Tag assoziiert: Der Tag `UnivTagOptions[1]` bestimmt das Verhalten von `TAG_BOOLEAN`, `UnivTagOptions[2]` das von `TAG_INTEGER`, usw.

Bei weniger als 30 Vektor-Elementen werden für die nicht spezifizierten Tags, wie auch für die mit einem leeren Vektor **«** als Option, die Defaults zum Codieren angewendet. Bei ungültigen Argumenten ist das Resultat ein leerer Vektor und `#.RCode` ist gesetzt.

Die Universal Tag Optionen

Zu den hier aufgeführten Tags der Class Universal kann die Art der Darstellung in den genesteten APL-Strukturen verändert werden.

Eine gewählte Einstellung betrifft stets beide Wandlungsrichtungen gleichermaßen, sowohl das Encodieren nach ASN.1, als auch das Decodieren nach APL.

Dieses anzupassen kann insbesondere dann erforderlich werden, wenn die APL-Datentypen nicht ausreichen, um den Wert eines ASN.1-Elements stets exakt darzustellen. So kennt etwa APL nur Integer-Werte bis 32 Bit, und als Real lassen sich längere ganzzahlige Werte ohne Genauigkeitsverlust auch nur bis 53 Bit abbilden. ASN.1 hingegen hat keine Begrenzung für die Bitlänge eines Integers.

Somit ist es nicht möglich, einen 54 Bit, oder wie in der Cryptographie alltäglich, gar einen 1024 Bit Integer in APL verlustfrei als numerischen Skalar zu handhaben.

Eine Darstellung als Character-Vektor bietet sich hier an. Diese Darstellung führt jedoch zu zusätzlichen Fragen: Sollen die noch exakt darstellbaren Integer weiterhin numerisch dargestellt werden, und welches Format erhält der Character-Vektor, direkter String, hexadezimal oder dezimal formatiert?

Daher können mit den Formulierungen:

```
UnivTagOptions←30⍴⊂⍬
UnivTagOptions[TAG_INTEGER]←UTO_HEX
RES←Depth UnivTagOptions Code ARG
```

oder:

```
RES←Depth (⍬ UTO_HEX) Code ARG
```

Code angewiesen werden, sämtliche Integer nur noch in hexadezimalem Format darzustellen und zu akzeptieren.

Des weiteren lässt sich durch die nachfolgende Option nahezu sämtlicher Wandlungs-Komfort abstellen:

```
RES←Depth (30⍴UTO_STR) Code ARG
```

Dieser Ausdruck bewirkt, dass Daten-Werte nur noch als direkte Character-Vektoren in APL erscheinen. Eine Ausnahme stellen nicht byte-bündige Bitstrings dar, die konzeptionell im APL-Format nur als boolesche Vektoren dargestellt werden können.

TAG_BOOLEAN (1)

Wert	Darstellung in APL
`UTO_STR` (0)	*Wert wird als direkter Character-Vektor dargestellt.*
`UTO_NUM` (1) ***default***	`FALSE` (0) oder `TRUE` (1)

TAG_INTEGER (2), TAG_ENUMERATED (10)

Wert	Darstellung in APL
UTO_STR (0) ***default***	Wert wird als direkter Character-Vektor dargestellt. Ist mit +UTO_I32, +UTO_I48 oder +UTO_I53 kombinierbar.
UTO_I32 (1)	Werte im Intervall [-2147483648 .. +2147483647] werden als numerischer Skalar dargestellt.
UTO_I48 (2) ***default***	Werte im Intervall [-140737488355328 .. +140737488355327] werden als numerischer Skalar dargestellt.
UTO_I53 (3)	Werte im Intervall [-9007199254740992 .. +9007199254740991] werden als numerischer Skalar dargestellt.
UTO_FMT (4)	Wert wird als dezimal formatierter Character-Vektor mit dem APL-Minuszeichen (¯) vor negativen Werten dargestellt. Akzeptiert werden führende und nachfolgende Leerzeichen, sowie +-¯ vor der Ziffernfolge. Ist mit +UTO_I32, +UTO_I48 oder +UTO_I53 kombinierbar.
UTO_HEX (8)	Wert wird als hexadezimaler Character-Vektor dargestellt. Führende und nachfolgende Leerzeichen, und auch , - zwischen den Zeichen paaren werden akzeptiert. Ist mit +UTO_I32, +UTO_I48 oder +UTO_I53 kombinierbar.

AG_BITSTRING (3)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt, sofern seine Länge ein Vielfaches von 8 ist. Nicht byte-bündige Bitstrings werden als boolesche Vektoren dargestellt.
UTO_NUM (1)	Wert wird stets als boolescher Vektor dargestellt.
UTO_SPCSEQ (2) ***default***	Wert wird als boolescher Vektor dargestellt. Enthält der Wert einen vollständig gültigen ASN.1-Inhalt, der mit SEQUENCE beginnt, wird dieser decodiert.
UTO_SPCALL (3)	Wert wird als boolescher Vektor dargestellt. Enthält der Wert einen vollständig gültigen ASN.1-Inhalt, der mit einem typischen Universal Tag beginnt, wird dieser decodiert.

TAG_OCTETSTRING (4)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_SPCSEQ (2)	Wert wird als direkter Character-Vektor dargestellt. Enthält der Wert einen vollständig gültigen ASN.1-Inhalt, der mit SEQUENCE beginnt, wird dieser decodiert.
UTO_SPCALL (3) ***default***	Wert wird als direkter Character-Vektor dargestellt. Enthält der Wert einen vollständig gültigen ASN.1-Inhalt, der mit einem typischen Universal Tag beginnt, wird dieser decodiert.

TAG_NULL (5)

Wert	Darstellung in APL
UTO_STR (0) ***default***	Bei jeder Länge ungleich Null im Header wird TAG_NULL als fehlerhaft identifiziert.

TAG_OID (6)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_NUM (1) ***default***	Wert wird als numerischer Vektor dargestellt.

TAG_NULL (5)

Wert	Darstellung in APL
UTO_STR (0) ***default***	Bei jeder Länge ungleich Null im Header wird TAG_NULL als fehlerhaft identifiziert.

TAG_OID (6)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_NUM (1) ***default***	Wert wird als numerischer Vektor dargestellt.

TAG_UTF8STR (12); TAG_ NUMERICSTR (18), TAG_PRINTABLESTR (19), TAG_T61STR (20) TAG_IA5STR (22), TAG_UNIVERSALSTR (28) TAG_BMPSTR (30)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_ANSI (1) ***default***	Wert wird als ANSI-Zeichenvektor der Systemdefault ANSI-Codepage dargestellt.
UTO_WIDE (2)	Wert wird als Wide-Character-Vektor im 2 Byte Little-Endian-Format der Unicode-Untermenge Basic Multilingual Plane dargestellt.

TAG_VIDEOTEXSTR (21), TAG_GRAPHICSTR (25) TAG_VISIBLESTR (26), TAG_GENERALSTR (27)

Wert	Darstellung in APL
UTO_STR (0) ***default***	Wert wird als direkter Character-Vektor dargestellt.
UTO_WIDE (2)	Wert wird als Wide-Character-Vektor im 2 Byte Little-Endian-Format der Unicode-Untermenge Basic Multilingual Plane dargestellt.

TAG_UTCTIME (23)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_ZULU (1)	*Datum und Zeit werden im ⍕TS-Format der Zeitzone Zulu/GMT dargestellt.* *Ist mit +UTO_AUTO kombinierbar.*
UTO_LOCAL (2) ***default***	Datum und Zeit werden im ⎕TS-Format der lokalen Zeitzone dargestellt. Ist mit +UTO_AUTO kombinierbar.
UTO_AUTO (4) ***default***	Beim Encodieren von Datumsangaben im ⎕TS-Format, deren Jahreszahl vor 1950 oder nach 2049 liegt, erfolgt eine automatische Wandlung auf TAG_GENERALIZEDTIME.

TAG_GENERALIZEDTIME (24)

Wert	Darstellung in APL
UTO_STR (0)	Wert wird als direkter Character-Vektor dargestellt.
UTO_ZULU (1)	Datum und Zeit werden im ⎕TS-Format der Zeitzone Zulu/GMT dargestellt.
UTO_LOCAL (2) ***default***	Datum und Zeit werden im ⎕TS-Format der lokalen Zeitzone dargestellt.

Nancy Wheeler

APL2 Namespaces

An Inside Look at the Journey from Mainframe to Workstation

We had: a well-documented design (APL2 Programming: System Services Reference, Packaged Workspaces by Michael Wheatley), a working implementation for mainframe APL2, Customer requirements for namespaces on workstation platforms and an internal goal to remove cross-system incompatibilities. We also had: APL2 Runtime Environment for Windows. We needed: a working implementation for workstation APL2, adherence to the existing design and co-existence with APL2 Runtime Environment.

At the end of the following „Journey from Mainframe to Workstation" we have: a working implementation for workstation APL2, close adherence to the existing design, full compatibility at execution time, minor differences in namespace creation, minor differences in invocation options and a few improvements over the mainframe implementation (i.e. variables not copied to active workspace and more flexibility in namespace creation). We are: one step closer to true cross-system compatibility.

Implementation Steps

Creation of Namespaces

Defining the File Properties

- **Mainframes:**
  - Workspace + Namespace Header + Magic = TEXT
  - TEXT + link = LOADLIB
  - Use standard operating system LOAD, shared segments
- **Workstations:**
  - Operating system formats not standard between Windows, Unix
  - APIs to manipulate DLLs not in Windows 95/98/ME
  - Link step tools not always available on customer machines
  - Workspace + Namespace Header = ANS
  - Lose: resource sharing capability
  - Gain: one-step, simpler creation process

Creating the File

- **Mainframes:**
  - `3 11 ⎕NA 'PACKAGE'`
  - `[namelist] PACKAGE wsfile`
  - `wsfile` is filename (no library numbers)
  - Link is separate step for user
- **Workstations:**
  - `PACKAGE` function name already in use by APL2 Runtime Environment
  - Term *package* ambigous due to APL2/PC use of term
  - `3 11 ⎕NA 'CNS'`
  - `[namelist] CNS wsfile [path]`
  - `wsfile` is `'filename'` or `[library number]wsname`
  - `path` specifies destination for namespace file
  - ➤ **Note:** workspace must be V2 workspace

Establishment of Associations

Processor 11 Syntax

- **Mainframes:**
  - `nameclass 11 ⎕NA 'NAME'`
  - `'(ddname)' 11 ⎕NA 'NAME'`
  - `'lib.member' 11 ⎕NA 'NAME'`
  - `'module' 11 ⎕NA 'NAME'`
  - `'' 11 ⎕NA 'NAME'`
  - Operating system facilities control file sharing, re-use
- **Workstations:**
  - `nameclass 11 ⎕NA 'NAME'`
  - `'(envvar)' 11 ⎕NA 'NAME'`
  - `'<namesfile>' 11 ⎕NA 'NAME'`
  - `'routine descriptor' 11 ⎕NA 'NAME'`
  - `'nsfile' 11 ⎕NA 'NAME'`
  - `'' 11 ⎕NA 'NAME'`
  - Read-only access allows multiple users of same file
  - File mapping used where available

Interpreter Issues

- **Re-resolution after)SAVE,)LOAD**
  - Occurs on first use, not on)LOAD
- **Re-resolution when namespace has changed**
  - Warning Message
  - Local Modifications Lost
- **Multiple names in same namespace**
  - P11 recognizes, keeps own use count
  - Interpreter also recognizes when P11 returns same address

- **Same name, different left arg**
- P11 would recognize if the two left args resolved to same file, but
- Interpreter does not call P11 since left args do not match
- **When to unload the namespace**
- Ideally, when last use is expunged
- Circular associations can cause confusion

Use of Namespace Objects

Mainframe/Workstation Compatibility High

- **Association and state maintained across)SAVE,)LOAD**
- **QNS, EXP external functions available**
- **Namescope-switching issues handled**
- Open defects address several edge conditions
- **Changed objects maintained in active ws**
- Mainframe: All variables copied permanently to active ws
- Workstation: Variables copied permanently to active ws only if changed
- Tradeoff: Space vs. Cycles
- **Vector Specification difference**
- Mainframe: External names not allowed
- Workstation: External names are allowed

Application Startup

Invocation Options

- **Mainframes:**
- RUN(function)
- Implies:

```
3 11 ⎕NA 'function'
function
```

- **Workstations:**
- -run already in use by APL2 Runtime Environment
- More flexibility desired in specification of namespace location
- -rns function[:locater]
- Implies:

```
'locater' 11 ⎕NA 'function'
function
```

- locater defaults to 3 if not specified

Kontakt

Nancy Wheeler, APL Products and Services, IBM Silicon Valley Laboratory, eMail: APL2@vnet.ibm.com

Anzeige

GUI-Utilities für Dyalog APL/W

Ein Buch von Kai Jäger, erschienen im Selbstverlag (BoD), erhältlich im Buchhandel und bei Internet-Anbietern wie Libri und Amazon mit der ISBN-Nr. 3-8311-1831-0.

Das Buch beschreibt frei verfügbare Software, mit deren Hilfe die ohnehin hohe Produktivität eines APL-Programmierers noch beträchtlich erhöht werden kann. Diese Software reduziert den Aufwand für die Erstellung eines GUIs erheblich, wenn die Aufgabenstellungen aus dem kaufmännischen Bereich stammen.

Das Buch wendet sich an den gestandenen APL-Programmierer, der Anwendungen realisiert. Für den APL-Neuling ist es nicht geeignet. Es ist insbesondere keine Einführung in die GUI-Programmierung mit APL. Es ist nur für Dyalog APL/W nutzbar.

Wer GUIs erstellen muss, der wird in der verfügbaren, außerordentlich mächtigen Software eine große Hilfe finden. Mit Hilfe dieses Buches ist auch die Einarbeitung in die grundlegenden Konzepte kein Problem mehr.

Wer APL als mathematisches Werkzeug für statistische Untersuchungen oder zur Datenanalyse benutzt, wird die hier vorgestellte Software nur dann sinnvoll nutzen können, wenn er zumindest gelegentlich auch GUIs erstellen muss.

Nebenbei lernt man noch frei verfügbare Software kennen, welche die Verwendung von Software-Modulen anderer Autoren in eigenen Applikationen stark vereinfacht. Auch Programmier-Teams werden durch diese Software unterstützt.

Weitere Informationen finden Sie unter:
http://www.kai-jaeger.de

www.kai-jaeger.de

Kai Jäger

Dyalog APL-Namespaces++: Dyalog Component Objects (DCOs)

Mit Hilfe von Namespaces können Code wie Variable gekapselt und zugleich strukturiert werden. Dies führt zu einem Code, der leichter zu erstellen, leichter zu verstehen und leichter zu warten ist. Zusätzlich besteht der Vorteil einer Kapselung darin, dass man „fremde" Namespaces gefahrlos verwenden kann, da sich die Gefahr von Namensüberschneidungen auf einen einzigen Namen oder nur auf einige wenige Objekte reduziert. Trotz aller Vorteile, die APL besitzt, erweist sich die fehlende Unterstützung von Programmierteams oftmals als eine Schwachstelle. Für ein Unternehmen, bei dem mehr als 50 Mitarbeiter mit Dyalog APL/W arbeiten, wurde deshalb eigens für Projektgruppen mit bis zu 15 Personen an einer Verbesserung gearbeitet. Als Ergebnis entstand eine Konvention, die definiert, welche Informationen ein Namespace enthalten muss oder sollte. Außerdem wurde Software für die Verwaltung solcher Namespaces realisiert, mit der auch größere Programmierteams unterstützt werden. Namespaces, die diese Konventionen einhalten, werden als DCOs („Dyalog Component Objects") bezeichnet.

Integrierte Dokumentation

Eine umfassende Dokumentation eines Software-Moduls gehört in der Regel *nicht* in den Code, aber die folgenden Dinge schon:

- Kurzbeschreibung, was das Modul tut.
- Listen der Methoden, Eigenschaften und ggf. Events.
- Angaben zum Autor samt Kontakt-Infos.
- Name der eigentlichen Dokumentation (Help-File, Word-Dokument, PDF-File...).
- Namen des Entwicklungs-Workspace und ggf. des Test-Workspace.
- Liste eventuell benötigter Software-Module (DLLs, EXE, DCOs...).
- Namen der „Init"- wie der „Cleanup"-Funktion.
- Typ (Generelles Modul? Projektbezogenes Modul? Welches Projekt?).
- Historie: was hat die aktuelle Version gegenüber der Vorgängerversion hinzugelernt?
- Fixes: welche Fehler wurden gefixt?

Versionsverwaltung

Unter „Versionsverwaltung" wird hier verstanden, dass der Programmierer durch den Vergleich eindeutiger, im jeweiligen Softwaremodul (DCO) enthaltener Informationen ermitteln kann, welches DCO das „beste" ist, wo sich diese Version befindet und was sich geändert hat. Außerdem muss es möglich sein, ein fertig gestelltes DCO automatisch zu verteilen, also in verschiedenen Verzeichnissen auf unterschiedlichen Festplatten zu speichern. Dabei sollte die Vorgänger-Version optional automatisch gesichert werden.

Für all diese Zwecke enthält ein DCO die folgenden Informationen:

- Alle DCOs verfügen über eine Versionsnummer wie z.B. 2.0 oder 3.2.
- Zusätzliche kann eine Releasenummer vergeben werden, die zwischen 1 und 99 liegen muss. Zusammen mit der Versionsnummer führt dies typischerweise zu 2.0.1 oder 3.2.16.
- Kennzeichnung als „Test"-Modul. Irgendwann ist ein DCO keine Baustelle mehr, muss aber noch getestet werden. Dann kennzeichnet der Autor das DCO als Test-Version. Auch der Zähler einer Testversion kann zwischen 1 und 99 liegen.

DCO-Manager

Die genannten Informationen sind die wichtigsten, die ein DCO zu bieten hat, vollständig ist diese Liste nicht. Alle diese Informationen und noch einige mehr können mit einer komfortablen Software gepflegt werden: dem DCO-Manager. Neben dem automatischen Ermitteln der besten Version und dem automatischen Verteilen eines neuen bzw. überarbeiteten DCOs bietet die Software noch Folgendes:

- Dialoge und Editoren für die Pflege der diversen Informationen.
- Vergleich des Codes zweier DCOs, etwa wenn ein DCO wieder zusammengeführt werden muss, das von zwei verschiedenen Programmierern geändert bzw. erweitert wurde.
- Erstellen eines Berichts: Welches DCO wurde mit welcher Versionsnummer wo gespeichert.

Komponenten-Files

Wie wir gesehen haben, ist ein DCO nichts anderes als ein „normaler" Namespace, angereichert mit bestimmten standardisierten Informationen. Um aber eine automatisierte Suche nach besten Versionen und ähnliches implementieren zu können, muss ein DCO in einem eigenständigen File gespeichert werden – als Bestandteil eines Workspace ließe sich dies nicht realisieren. Hierfür werden die APL-typischen „Komponenten-Files" verwendet, wie sie in jedem APL-System mit Ausnahme von APL2 zur Verfügung stehen. In einem Komponenten-File kann eine beliebige Anzahl von Komponenten abgespeichert werden. Jede Komponente kann wiederum ein beliebiges APL-Array enthalten. Beliebig heißt: beliebige Datentypen, beliebige Schachtelungstiefen, beliebige Größen – jedenfalls fast beliebig: Restriktionen gibt es natürlich, aber sie werden normalerweise nicht bemerkt.

Nun ist ein Namespace aber kein Array, kann also nicht einfach in einer Komponente abgespeichert werden, aber Dyalog APL kennt eine System-Funktion ⎕OR (Object Representation), mit der ein Namespace in ein Array verwandelt werden

kann. Dieses Array kann dann sehr wohl in einem Komponenten-File gespeichert werden. Genauso wichtig: das Array kann mit []NS oder []WC wieder in einen Namespace verwandelt werden. Diese Operationen sind in beide Richtungen schnell, weil nur ein einziges Bit umgeschaltet wird.

Eine wichtige Eigenschaft von DCOs ist also, dass es sich um Namespaces handelt, die nicht in Workspaces, sondern in Komponenten-Files gespeichert werden. Alle DCO-spezifischen Zusatzinformationen werden in einem Sub-Namespace „DCO_Info“ zusammengefasst. Heißt unser Namespace „XYZ“, dann enthält er einen Namespace „XYZ.DCO_Info“. Wird ein DCO in ein Komponenten-File geschrieben, so wird der Inhalt des Namespace „DCO_Info“ in einer eigenen Komponente gespeichert. Dies erlaubt den schnellen Zugriff auf alle DCO-spezifischen Informationen. Dies gilt insbesondere auch dann, wenn das „eigentliche“ DCO sehr groß ist.

Wozu DCOs?

Namespaces bieten sich natürlich geradezu an, bestimmte Funktionalitäten nach Gruppen zusammenzufassen, entsprechend zu dokumentieren, auf Datei zu speichern und anderen Programmierern zur Verfügung zu stellen. In der Tat gibt es mittlerweile eine beträchtliche Anzahl von DCOs für die unterschiedlichsten Einsatzzwecke. Von der Web-Site der APL Software Team GmbH (www.aplteam.de) kann nicht nur der freie DCO-Manager heruntergeladen werden, es stehen auch eine ganze Reihe von DCOs zum Download zur Verfügung, deren Nutzung nur einigen wenigen lizenzrechtlichen Restriktionen unterliegt. Nach Kategorien zusammengefasst warten dort auf den Download:

- Mehr als 10 DCOs für die schnelle Erstellung diverser Dialoge (GUI-Programmierung).
- 2 DCOs für die Verwendung von TCP/IP, eines für UDP, eines für TCP.
- Lesen von und Schreiben in die Windows Registry.
- Anzeige von HTML-Help-Files aus Dyalog APL/W heraus.
- Umgang mit Dateien: Kopieren, Löschen, Umbenennen und Verschieben, das aktuelle Verzeichnis abfragen wie setzen und vieles mehr.
- Erstellen von Log-Files.
- Import von APL2-Workspaces (Host oder PC).
- Lesen und Schreiben von Komponenten-Files mit Schlüsseln.
- Drucken – simpel und einfach.
- Generelle Werkzeuge (#.wtutil)

Wie werden DCO verwendet?

Es gibt grundsätzlich zwei verschiedene Möglichkeiten:

- Laden eines oder mehrerer DCOs interaktiv durch den DCO-Manager. Die so geladenen DCOs werden als Bestandteil des Workspace gespeichert. Dies leistet der DCO-Manager.
- Laden eines oder mehrerer DCOs per Programm. Für diesen Zweck bietet das DCO „#.wtutil“ die Methode „Include_DCO“.

Eine ausführliche Besprechung der GUI-orientierten DCOs findet sich in der Neuerscheinung „GUI-Utilities für Dyalog APL/W“. Nähere Infos bei http://www.kai-jaeger.de oder bei Amazon und anderen. Dieses Buch enthält auch eine ausführliche Besprechung von DCOs sowie des DCO-Managers. Wer sich für DCOs interessiert, aber nicht für die GUI-DCOs, der wird in Kürze ein PDF-Dokument von der gleichen Web-Site herunterladen können, das sich ausschließlich mit DCOs beschäftigen wird.

DCOs nutzen– ein Beispiel

Immer häufiger wird in modernen Anwendungen TCP/IP verwendet, um mit der Außenwelt zu kommunizieren. Eine solche Kommunikation mit den in Dyalog APL/W eingebauten Bordmitteln zu realisieren, ist recht einfach – wenn alles einwandfrei funktioniert! Ein wenig schwieriger ist es, Sonderbedingungen und Fehlerbehandlungen einzubauen sowie leistungsfähige Protokoll-Funktionen zu realisieren. All dies und noch einiges mehr bietet das DCO „TCP“ der APL Software Team GmbH. (Downloadmöglichkeit: www.aplteam. de/toc/hp2down/overview_d.htm)

Laden Sie aber nicht nur das DCO, sondern auch das Help-File herunter. Die ZIP-Files packen Sie anschließend bitte aus.

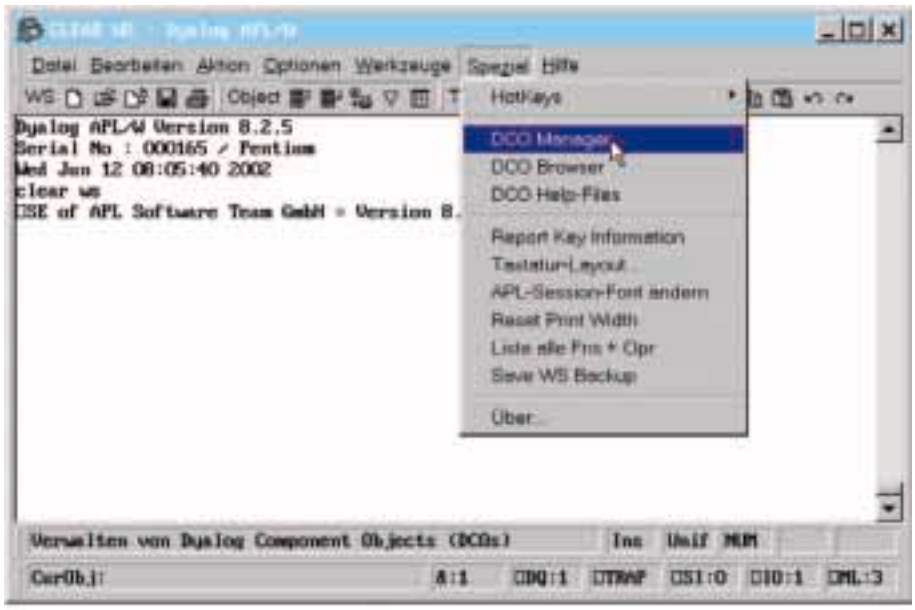
Um ein DCO nach dem Download verwenden zu können, benötigen Sie außerdem den DCO-Manager. Am einfachsten ist es, den Workspace APLST82.DWS bzw. APLST90.DWS (die Zahlen stehen für die verwendete APL-Version) von „www.aplteam.de/hp2downqse_ d.htm“ herunter zu laden, zu öffnen und die Anweisungen im []LX zu befolgen. Vergessen Sie nicht, anschließend das Session-Objekt zu speichern.

Ziel

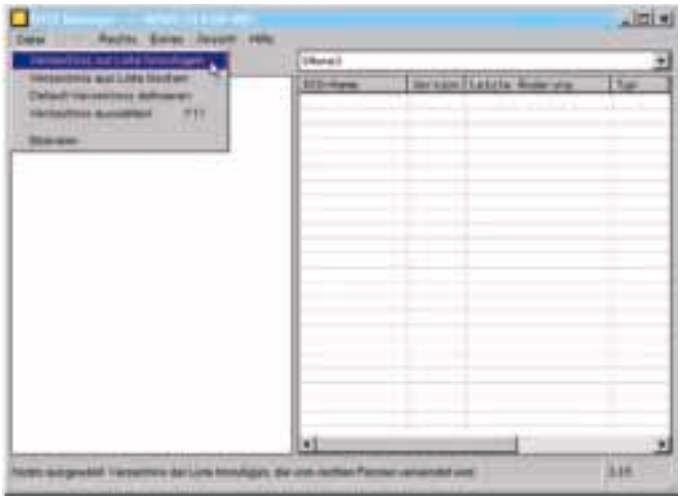
Nachfolgend wird am Beispiel des DCOs „TCP“ untersucht, wie man den DCO-Manager benutzen kann. Allerdings kann man hier nur an der Oberfläche bleiben – eine detaillierte Beschreibung des Leistungsumfangs des DCO-Managers, wie sie in dem Buch „GUI-Utilities für Dyalog APL/W“ enthalten ist, umfasst knapp 60 Seiten. Die Software kann somit erheblich mehr, als hier besprochen wird.

Start

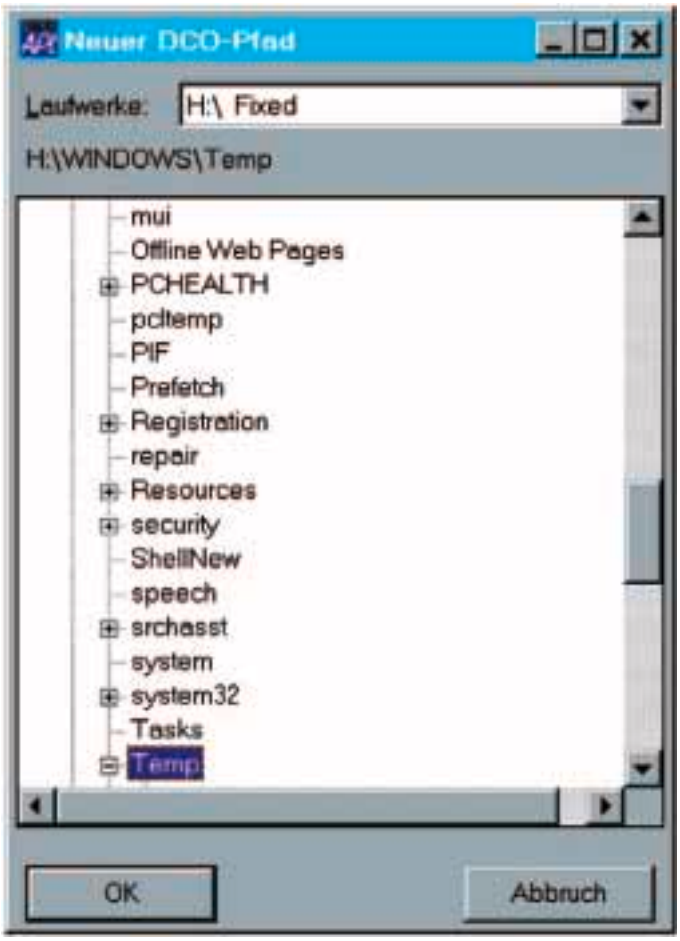
Nachdem das Session-Objekt mit dem DCO-Manager verfügbar ist, können wir den DCO-Manager mit dem Befehl „DCO-Manager“ aus dem neuen Menü „Spezial“ starten:



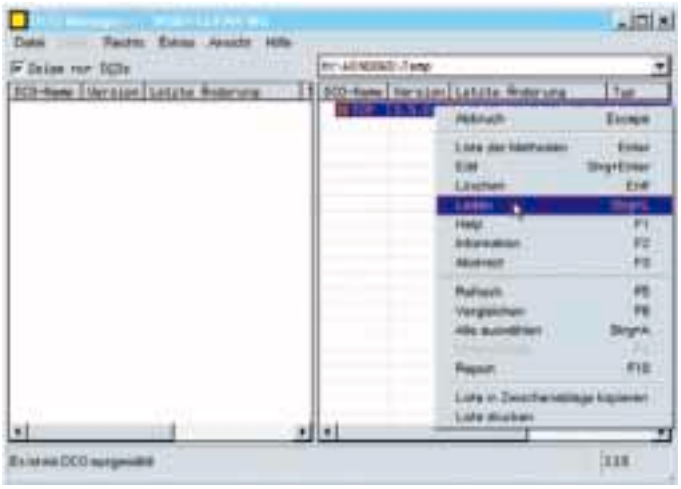
In dem folgenden Fenster "schauen" Sie mit dem linken Teil-Fenster in Ihren Workspace, mit dem rechten auf Ihre Festplatte. Wir müssen den DCO-Manager nun dazu bringen, in das Verzeichnis zu schauen, in welches Sie das DCO "TCP" nach dem Download gespeichert haben. Rufen Sie dafür den Befehl "Verzeichnis zur Liste hinzufügen" aus dem Menü "Datei" auf:



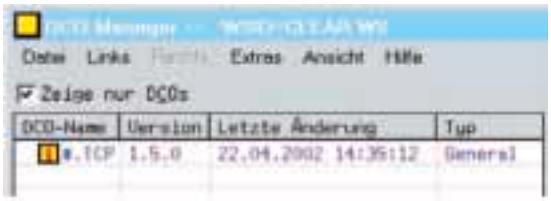
Es öffnet sich ein Dialog, der Ihnen die Auswahl des Verzeichnisses erlaubt, in das Sie das DCO "TCP" sowie das dazu gehörende Help-File abgelegt haben:



Dieser Dialog ist mit dem DCO "DirSelect" realisiert. Sobald Sie diesen Dialog beenden, zeigt der DCO-Manager dieses Verzeichnis rechts oben in der Combo-Box an. Als einziges DCO wird "TCP" gezeigt. Klickt man dieses DCO mit der rechten Maustaste an, kann man anschließend aus dem Kontext-Menü den Befehl "Laden" auswählen.

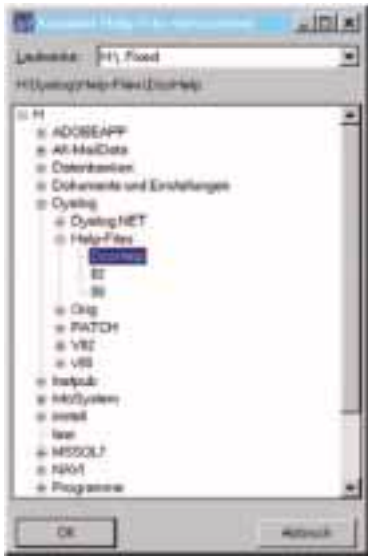


Das DCO "TCP" wurde aus dem Komponenten-File in den aktuellen Workspace geladen, daher wird es jetzt im linken Fenster angezeigt - es kann jetzt benutzt werden:



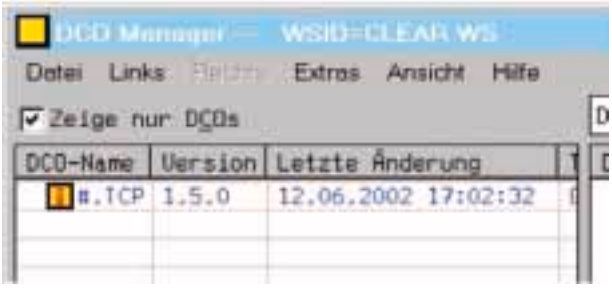
Beachten Sie, dass der DCO-Manager ein gelbes Viereck mit einem roten Ausrufezeichen zeigt – auf Ihrer Maschine wird das rote Ausrufezeichen fehlen. Es zeigt die Verfügbarkeit eines Help-Files an, das zu dem DCO gehört. Zwar werden Sie dieses Help-File heruntergeladen haben, aber der DCO-Manager kann es (noch) nicht finden. Rufen Sie den Befehl "Hilfe|DCO-Help-Files" auf:

In dem sich öffnenden Fenster klicken Sie sich zu dem Verzeichnis durch, in dem Sie das Help-File abgelegt haben:

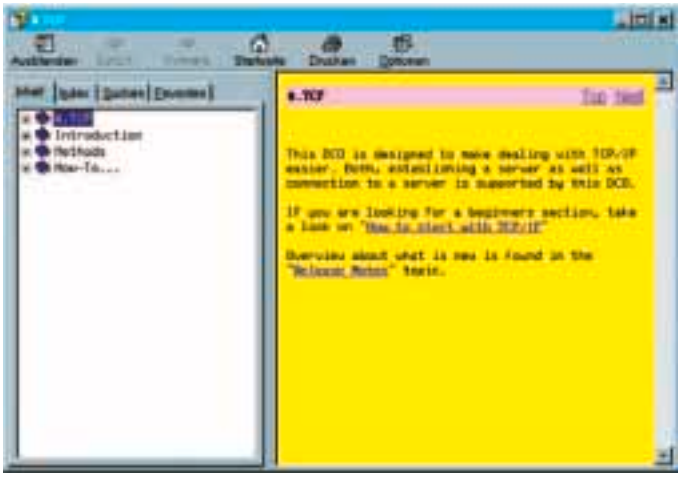


Anschließend werden Ihnen in einem Auswahl-Fenster alle Help-Files präsentiert, die in dem von Ihnen ausgewählten Verzeichnis gefunden werden konnten. Allerdings sehen Sie nur Help-Files des Typs „kompiliertes HTML“, das sind Dateien mit der Endung „chm“. Wir wollen das Help-File jetzt nicht starten, sondern einen anderen, einfacheren Weg benutzen; wir verlassen diesen Dialog daher mit <Escape>.

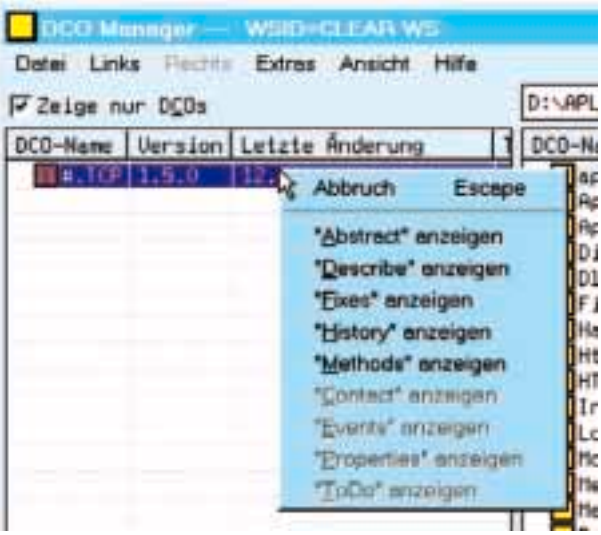
Wählen Sie jetzt im DCO-Manager den Befehl „Links | Refresh“ aus oder drücken Sie F5. Der DCO-Manager erkennt daraufhin, dass in dem Help-File-Verzeichnis ein Help-File für das DCO „TCP“ vorliegt und stellt dies auch dar:



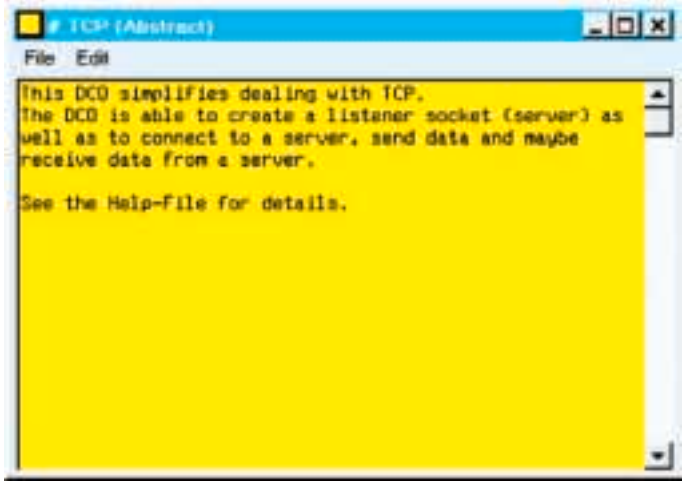
Wird das DCO angeklickt (selektiert) und dann F1 gedrückt, wird das Help-File geöffnet:



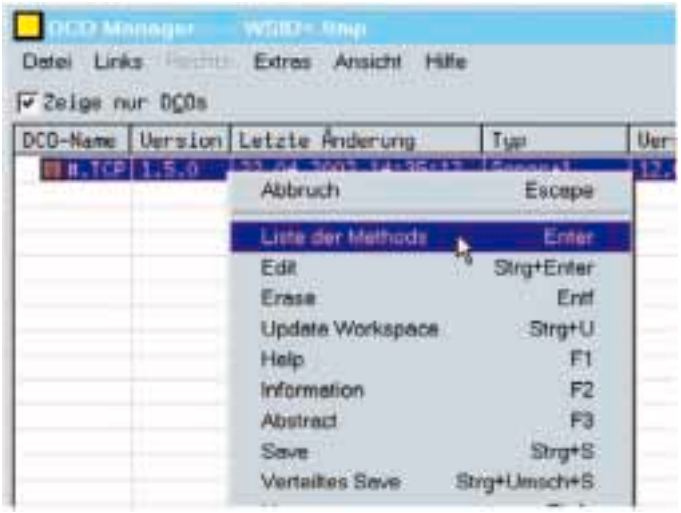
Drückt man statt dessen F2, werden in einem Popup-Menü alle potentiell verfügbaren Zusatz-Infos angeboten. Die für das DCO „TCP“ nicht vorhandenen Informationen bleiben grau:



Wählt man „Abstract“ aus, wird ein Fenster geöffnet, mit einer kurzen Beschreibung, was das ausgewählte DCO macht:

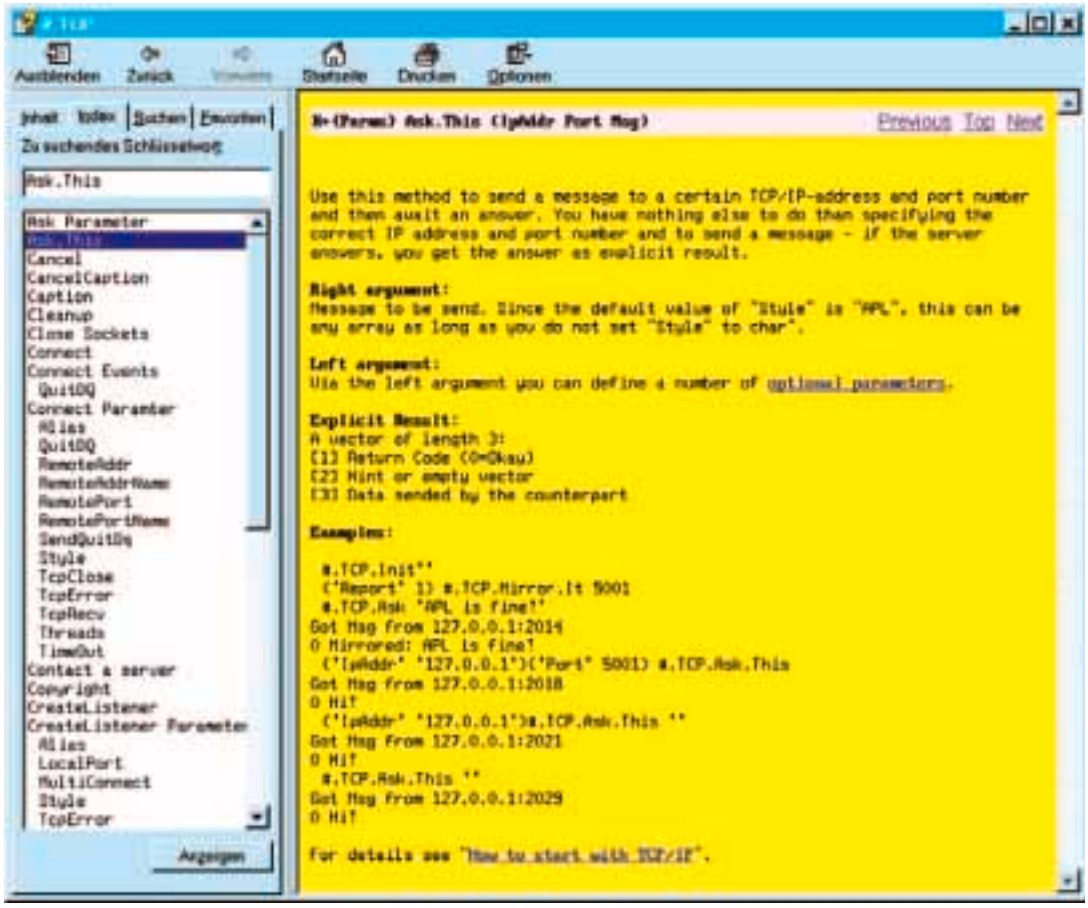


Eine für die Nutzung der weitaus meisten DCOs sehr wichtige Information wird in dem Kontextmenü unter „Liste der Methods“ angeboten:



Hier sehen Sie die Liste aller für das DCO „TCP“ definierten Methoden. Über das Kontextmenü stehen eine Reihe wichtiger Befehle zur Verfügung:

Die im Kontextmenü angebotene Hilfe für eine bestimme Methode startet das Help-File und schlägt sofort die Seite für die gewählte Methode auf:



Wie Sie sehen: es ist es einfach, an eine Dokumentation der eingebauten Methoden zu gelangen.

Um das DCO zu untersuchen, werden wir zwei Workspaces laden. In dem einen Workspace wird ein Listener-Objekt eingerichtet. Ein Listener „horcht" an einem bestimmten Port auf Kontaktversuche. Inder Regel müssen Sie sich selbst darum kümmern, was mit einem solchen Kontaktversuch geschehen soll, das heißt einen Handler für den Event „TcpRecv" implementieren. Es gibt für Testzwecke aber eine Methode „Mirror.It", die dies für uns übernimmt. Diese Funktion benötigt nur einen einzigen Parameter: die Nummer des zu überwachenden Ports. Die Methode „Mirror.It" nimmt an diesem Port eine Message entgegen, stellt ihr den String „Mirrored:" voran, hängt „<CR><LF><CR><LF>" an – und schickt das Ergebnis an den Client zurück.

Weil „Mirror.It" für Testzwecke gedacht ist, wird der Parameter „Report" standardmäßig auf 1 gesetzt. Dies veranlasst „Mirror.It" zur Ausgabe diverser Meldungen im Session Manager.

Bevor wir aber das DCO „TCP" benutzen können, müssen wir das DCO zunächst initialisieren. Dies geschieht mit:

```
#.TCP.Init''
```

Nun rufen wir "Mirror.It" auf:

```
#.TCP.Mirror.It 5001
```

Dies veranlasst Mirror.It, den Port 5001 zu überwachen. Nun starten wir ein zweites APL, laden das DCO "TCP" mit Hilfe des DCO-Managers auch in den zweiten Workspace und initialisieren das DCO:

```
#.TCP.Init''
```

Jetzt können wir eine Message an den Server senden. Wir wissen, dass der Server unsere Message leicht verändert wieder an uns zurückschicken wird. Wir benötigen also nicht nur eine Methode, die eine Message an eine Zieladresse versendet, wir müssen auch Vorkehrungen treffen, um die Antwort zu empfangen. Für diesen Zweck gibt es die fertige Methode "#.TCP.Ask.This". Diese Methode benötigt zwingend die zu sendenden Daten, und wenn dies nur ein leerer Vektor ist. Diese werden als rechtes Argument an "Ask.This" übergeben.

Außerdem benötigt "Ask.This" in der Regel noch zwei weitere, aber dennoch optionale Parameter:

- **IpAddr**
  Die IP-Adresse des Servers. In unserem Fall ist dies „127.0.0.", das TCP/IP-Standard-Kürzel für „meine eigene Maschine". Da genau dies der Standardwert für den optionalen Parameter „IpAddr" ist, können wir uns in diesem Fall die Übergabe des Parameters sparen.

- **Port**
  Die Portnummer. Muss natürlich die Portnummer sein, an welcher der Server „lauscht", sonst kann es nicht funktionieren. Auch für die Portnummer verwendet „Ask.This" einen Default, nämlich 5001 – daher müssen wir auch die (optionale) Portnummer nicht definieren, denn genau diese Portnummer haben wir ja auch beim Server verwendet.

Im zweiten Workspace geben wir also ein:

```
#.TCP.Ask.This 'APL is fine!'
```

```
0   Mirrored: APL is fine!
```

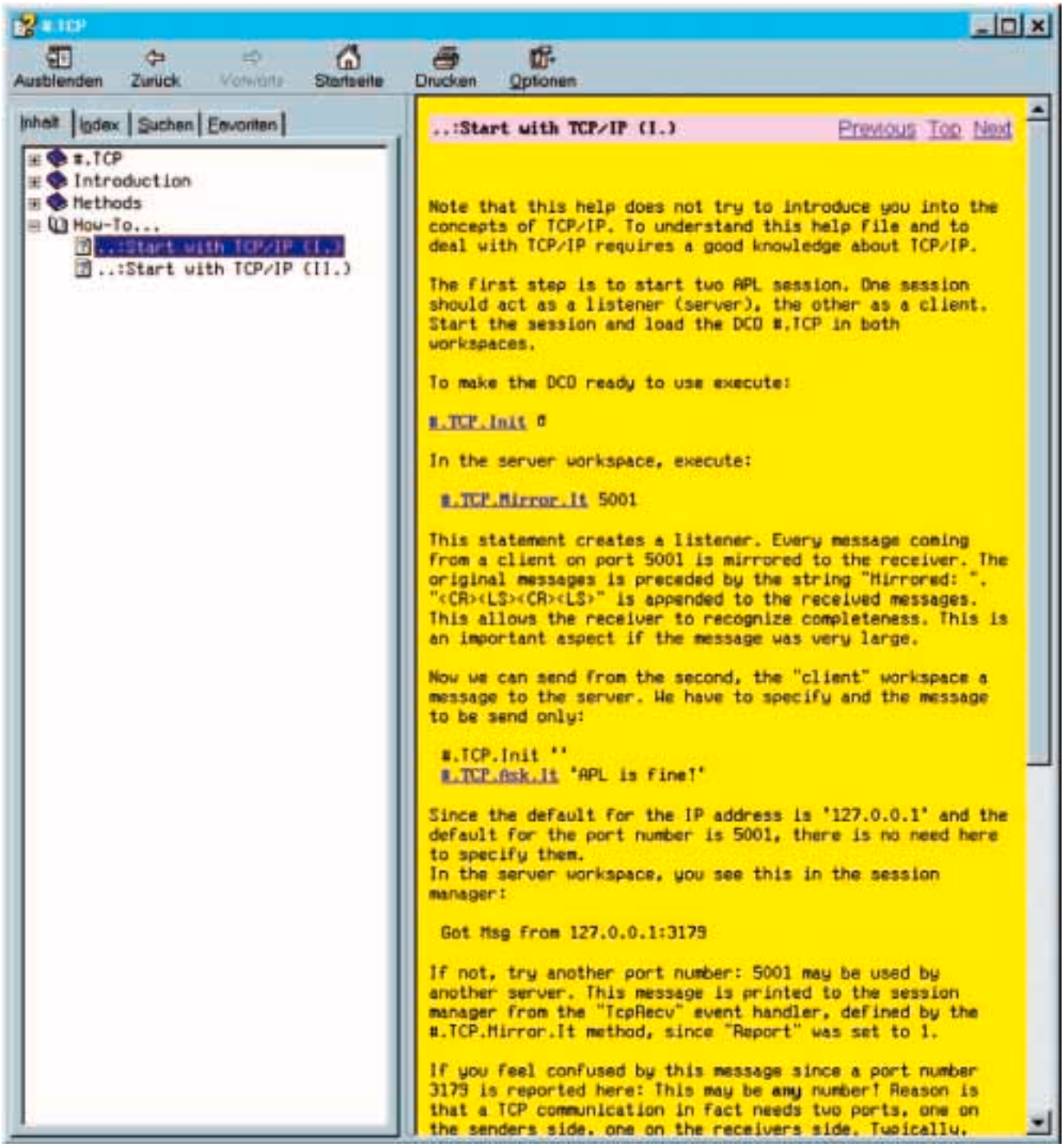
Die zweite Zeile ist das explizite Ergebnis der Methode "#.TCP.Ask.This": ein Return-Code, hier 0, ein Hint, hier ein leerer Vektor, sowie die vom Server gesendeten Daten. In dem zuerst geladenen APL, unserem "Server", sehen wir dies:

```
Got Msg from 127.0.0.1:3027 "APL is fine!"
```

Es kann sein, dass es bei Ihnen *nicht* funktioniert – dann probieren Sie es noch einmal mit einer anderen Portnummer: die Num-

mer 5001 könnte auf Ihrem Rechner bereits vergeben sein, obwohl dies unwahrscheinlich ist. Wenn auf Ihrem Rechner TCP/IP korrekt installiert ist, werden ansonsten keine Probleme auftreten.

Man kann sich den Zugang zu einem DCO wie „TCP“ auch ohne jedes Wissen erarbeiten: das Help-File enthält alle benötigten Informationen. Wie fast alle für DCOs angebotenen Hel-Files enthält auch das TCP-Help-File eine „How-To...“-Sektion:



Kontakt

Kai Jäger (freier EDV-Berater)
Eibenstraße 1, D-90513 Zirndorf, Tel. 09911- 482512, Fax: 0911 -482518, eMail: kai@kai-jaeger.de, Internet: www.kai-jaeger.de.

Wolfgang Strasser

Miszellen aus der Finanzmathematik

Es wäre sicherlich eine schöne Sache, wenn man aufgrund der Vergangenheitsdaten eines Aktienportfolios dessen zukünftige Entwicklung voraussagen könnte. Dies erscheint jedoch schwierig, wenn nicht unmöglich. Denn schliesslich sind auch überzeugte «Chartisten» durch Spekulation noch nicht reich geworden, obwohl sie doch vorgeben, ihre geheimnisvolle Lehre sei die „einzig wahre". Wenn keine Geheimlehre hilft, dann könnte vielleicht die Wissenschaft weiterhelfen. Im folgenden Beitrag werden interessante Ansätze aus der Finanzmathematik vorgestellt, die mit Hilfe von APL-Programmen jeder am Computer nachprüfen kann.

Die Aktienmarkttheorie, auf die in Kapitel 2 eingegangen wird, sagt eindeutig, dass die Vergangenheitsdaten wertlos sind und die zukünftigen Kurse allein vom inneren Wert der Aktien, also von ihrer «Drift» μ, abhängen, sowie von gedächtnislosen Zufallsschwankungen, die noch mit einem Faktor s skaliert werden. Da sich die zufälligen Abweichungen über die Zeit zu 0 addieren, verfügen wir damit über eine Formel, die den *erwarteten*, zukünftigen Kurs aufgrund der alleinigen Kenntnis von μ eindeutig festlegt.

Das Konfidenzintervall um diesen Erwartungswert kann sehr gross sein. Es hängt wesentlich davon ab, wie gross die Ausschläge der skalierten Zufallsbewegung sind. Also muss man den «Random Walk» studieren. In Kapitel 3 wird die berühmte Brownsche Bewegung konstruiert, da sie nach herrschender Meinung die Zufallsschwankungen von Aktien beschreibt. Man wird sehen, dass es sich dabei um nichts Geheimnisvolles handelt, sondern um einen sehr simplen Prozess, den jeder leicht auf dem Computer nachvollziehen kann.

In Kapitel 4 wird schliesslich die Kritik vorgestellt, die vornehmlich Mandelbrot an der Brownschen Bewegung als Modell für Aktienschwankungen übt. Die von ihm als Ersatz vorgeschlagenen Bewegungen aus der Klasse der Lévy-stabilen Verteilungen werden dort ebenfalls beschrieben und konstruiert.

Vielleicht befürchtet der Leser jetzt – nicht ganz zu unrecht –, dass all die Theorie ihm doch nicht die «garantiert richtige» Kaufempfehlung liefert. Daher sei ihm zumindest ein Instrument an die Hand gegeben – jetzt und sofort – mit dem er drohende Verluste durch die Investition in ein Aktienportfolio begrenzen oder gar vermeiden kann. Dieses Instrument besteht darin, dass man sich gegen einen festen Euro-Betrag bei seiner Hausbank Optionen auf sein Aktienportefeuille kauft.

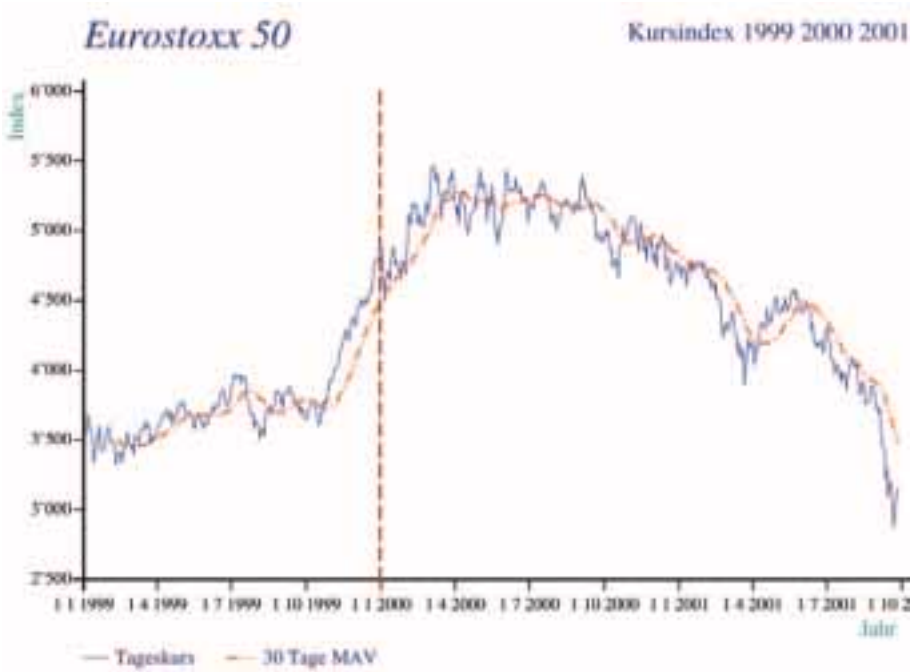


Abbildung 1: Kursindex 1999 bis 2001

1 Wie berechnet sich der Preis einer Option?

Die Berechnung von Optionspreisen ist eng mit den Namen F. Black und M. Scholes verknüpft. Black und Scholes[1] haben für ihren Artikel zur Optionsbewertung den Nobelpreis für Wirtschaftswissenschaften des Jahres 1997 zusammen mit Robert C. Merton[2] erhalten.

Ausgangspunkt ihrer Überlegungen ist das Modell der Investmentprozesse, das in Kapitel 2 vorgestellt wir. Sodann benötigen sie recht komplizierte Mathematik, die von Norbert Wiener[3] und Kiyosi Itô[4] entwickelt wurde, übrigens ohne den geringsten Bezug zur Finanzwelt. Aus diesem Mix wurde das Black & Scholes Modell der Optionspreise geboren. Das Modell wird, leicht verfeinert oder abgewandelt, von allen Banken und Finanzinstituten der Welt als Standardinstrument eingesetzt.

In der Welt der Optionen unterscheidet man begrifflich die komplementären Positionen «Käufer» (long position) und «Verkäufer» (short position) sowie die Optionsart «PUT», die dem Käufer das Recht einräumt, sein Portfolio (underlying) zu *verkaufen*, und die Optionsart «CALL», die dem Käufer das Recht gibt, ein Portfolio zu *kaufen*.

Nehmen wir an, ein Interessent wolle eine PUT Option für sein Portfolio kaufen. Er erwirbt damit das Recht, sein Portfolio zum vereinbarten Fälligkeitstermin (dann und nur dann = europäische Option) zu einem vereinbarten Preis (Basispreis, exercise price) zu verkaufen.

1 F. Black & M. Scholes, The pricing of options and corporate liabilities, J. Pol. Economy, 3, 1973

2 R.C. Merton, Continuous-time Finance, Basil & Blackwell, 1990

3 N. Wiener, Differential Space, JMP,1923

4 K. Itô, Stochastic Integral, Proceedings Imperial Academy Tokio, 1944

Der Preis für eine solche Option berechnet sich nach der Black & Scholes Formel:

$$P = B \cdot e^{-rf \cdot T} \cdot N(-d_2) - K \cdot N(-d_1)$$

mit :

$$d_1 = \frac{\ln K/B + (rf + 0.5 \cdot \sigma^2) \cdot T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln K/B + (rf - 0.5 \cdot \sigma^2) \cdot T}{\sigma\sqrt{T}}$$

Die hier benutzten Variablen haben nachstehende Bedeutung und werden im Beispiel mit Zahlen belegt:

K = 100.0	Aktienkurs zu Zeitpunkt T0, current stock price
B = 105.5	Basispreis bei Fälligkeit; exercise price
T = 0.75	(Rest-)Laufzeit in Jahren (hier ¾ Jahr)
s = 0.20	Erwartete Volatilität des Aktienkurses p.a.
rf = 0.02958...	Risikoloser Zins p.a. bei stetiger Verzinsung (also hier: ln 1.03)
N(a)	Standard-Normalverteilung (Integral Funktion)

Die Übergabe der Parameter an die Funktion BSput liefert einen Preis von € 8.72 als Ergebnis:

```
      BSput 100   105.5   0.75   0.2   1.03
8.719038446
```

Das Programm BSput wird am Endes dieses Artikels wiedergegeben. Es nutzt die Black-Scholes Formel für CALL-Optionen und konvertiert anschliessend unter Ausnutzung der Komplementarität von Put- und Call-Optionen (Put-Call-Parität). Der Leser erhält damit alle drei Black-Scholes-Formeln.

2 Der Investment Prozess

Das vorherrschende Modell des Investment Prozesses geht auf Louis Bachelier[5] zurück. Im Jahr 1900 schrieb dieser seine These mit dem bezeichnenden Titel «Théorie de la spéculation», der grundlegend für die Finanzmathematik wurde. Nach diesem Modell hängt der zukünftige Aktienwert S(t) zum Zeitpunkt t vom Ausgangswert S(0) ab, sodann von der Drift, also dem langfristigen Erwartungswert μ der Rendite, sowie von einer zufälligen Störvariablen W(t), die mit der Streuung σ (Volatilität) skaliert wird:

$$S(t) = S(0) \cdot (\mu \cdot t + \sigma \cdot W(t))$$

Ist die Störvariable W(t) eine standardnormalverteilte N(0,1) Zufallsvariable, so können wir unsere ursprüngliche Fragestellung nach dem zukünftigen Wert einer Aktie exakt beantworten: Der Wert S(t) hängt nur vom inneren Wert der Aktie ab, und alle aufeinander folgenden Preisänderungen der Vergangenheit sind statistisch unabhängig. Damit wird auch jeglichem Chartismus widersprochen; schliesslich haben die Vergangenheitswerte kein «Gedächtnis».

Da das Modell gemäss obiger Gleichungen auch negative Preise für den Stock Price S(t) zulässt (was wenig realistisch ist), es darüber hinaus nicht invariant ist gegenüber Währungseinheiten und es letztlich nicht der intrinsischen Annahme einer permanenten Wertsteigerung unserer Wirtschaft entspricht, hat Bachelier eine exponentielle Variante als «korrektes» Modell nachgereicht:

$$S(t) = S(0) \cdot \exp\{(\mu \cdot t + \sigma \cdot W(t))\}$$

Legen wir dieses allgemein anerkannte Modell zugrunde – es heisst jetzt auch Black-Scholes-Modell – , so ist S(t) lognormalverteilt und der Erwartungswert des Kurses zum Zeitpunkt t ist gerade:

$$E((S(t))) = S(0) \cdot e^{\mu \cdot t}$$

Ist die Volatilität von S bekannt, so lässt sich auch ein Konfidenzintervall für den Erwartungswert angeben. Für den DAX wird daher seit 1994 von der Deutschen Börse ein Volatilitätsindex, der VDAX, veröffentlicht. Man sieht, die Sache ist ernst.

3 Der Wiener Prozess / die Brownsche Bewegung

Bei der Störvariablen W(t) handelt es sich um die Brownsche Bewegung oder auch um einen Wiener Prozess (nach Norbert Wiener). Dieses Phänomen hat viele Namen und viele Väter, die unabhängig voneinander auf die Verteilung aufmerksam wurden.

Der erste war wohl der schottische Botaniker Robert Brown, der sie um 1850 im Zusammenhang mit Pollenbewegungen in Flüssigkeit beschrieben hat und erst viel später auch ihr Namensgeber wurde. Als Nächster beschrieb Bachelier in o.g. These die Brownsche Bewegung, ohne Kenntnis allerdings von Mr. Brown. In der Physik tauchte die Frage nach einem Modell für Molekularbewegungen auf, die Albert Einstein[6] 1905 und 1906 ebenfalls mit Hilfe derselben Verteilung beschrieb, ohne seinerseits von Brown oder Bachelier zu wissen. Schliesslich formalisierte N. Wiener den stochastischen Prozess (aus Physikersicht) im Jahre 1923.

Die Anforderung an die Zufallsbewegung (Random Walk) sind ganz simpel: In einem festen Zeitintervall soll die Funktion N Bewegungen ausführen. Bei Aktienkursen erfolgt die Bewegung entlang nur einer Dimension, entweder nach oben, oder nach unten. Bei Teilchen wird man verlangen, dass sie sich in eine beliebige Richtung in der Ebene oder im Raum bewegen können.

Formal können wir die Zufallsbewegung als binomial verteilt für alle positiven ganzen Zahlen n mit folgender Regel definieren:

- $W_n(0)=0$
- Abszisseneinteilung: 1/n
- Wahrscheinlichkeit eines Sprungs nach oben oder unten: je 50%
- Sprunghöhe nach oben (+1) und unten (-1): $\pm 1 \times 1/\sqrt{n}$

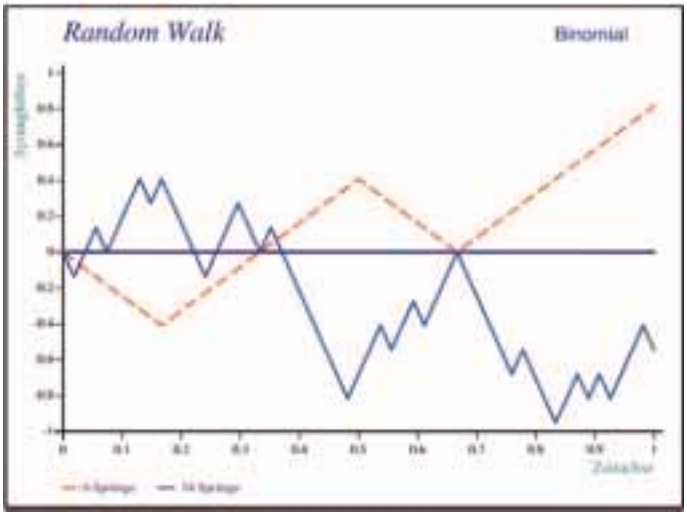


Abbildung 2: Zufallsbewegung mit nur wenigen Sprüngen

5 L. Bachelier, Théorie de la spéculation, théorie mathématique du jeu, Ann. Sc. ENS, 1900

6 A. Einstein, Annalen der Physik, 17, 1905

Die Definition kann einfach in die APL Notation umgesetzt werden. Die Funktion «brown» besteht aus den drei wesentlichen Schritten:

```
[1]  a ← uniform N              ⍝ N gleichverteilte ZV generieren
[2]  Xj ← (a>0.5)+¯1×a≤0.5      ⍝ ZV auf ¯1 & 1 konzentrieren
[3]  W ← 0,+\Xj+N*0.5           ⍝ Start 0, aufsummieren und div.N*1/2
```

Bei nur wenigen Sprüngen (siehe Abbildung 2) ist man intuitiv noch etwas von Aktienkursbewegungen entfernt; deutlich besser wird der Eindruck, wenn man die Anzahl der Bewegungen erhöht (siehe Abbildung 3):

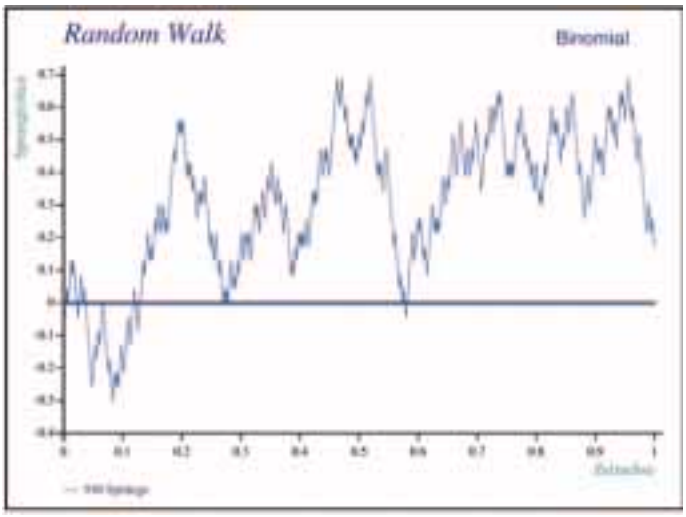


Abbildung 3: Zufallsbewegung mit mehreren Sprüngen

Nun kann man ausnutzen, dass für binomial verteilte Variablen der zentrale Grenzwertsatz gilt, d.h. dass die Verteilung W_n (1) für grosse n gegen die Standard-Normalverteilung N(0,1) strebt. Dann gilt für $W_n(t)$, dass diese Folge gegen N(0,t)=N(0,1)×$t^{1/2}$ strebt:

$$W_n(t) = \sqrt{t}\left(\frac{\sum_{j=1}^{nt} X_j}{\sqrt{nt}}\right)$$

Die graphische Darstellung der Brownschen Bewegung sieht jetzt schon recht beeindruckend aus (siehe Abbildung 4):

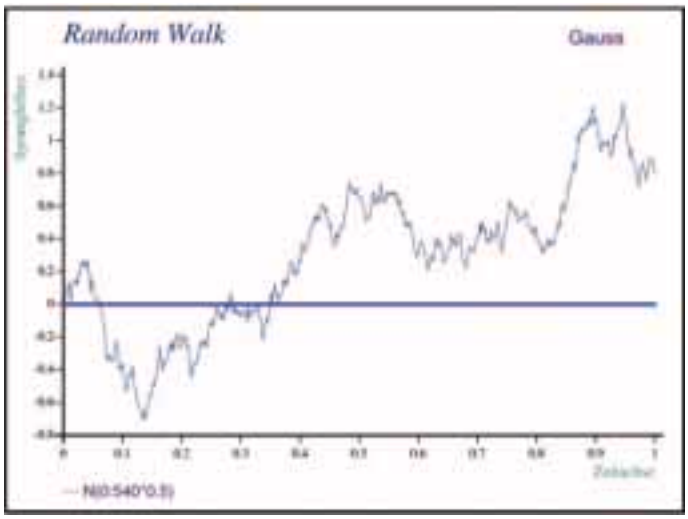


Abbildung 4: Darstellung der Brownschen Bewegung

Formal erhalten wir die Definition:

Der Prozess W ist eine Brownsche Bewegung, wenn gilt

- W(t) ist stetig und W(0)=0
- W(t) ist normalverteilt mit N(0,t)
- Der Zuwachs W(t+d)–W(t) ist normalverteilt mit N(0,t) und unabhängig von d

Für das exponentielle Aktienmodell nach Bachelier ergibt sich somit, dass die Inkremente der logarithmierten Aktienkurse einer Brownschen Bewegung folgen.

$$\frac{\ln S(t+d) - \ln S(t)}{\sqrt{d}}$$

Zum Generieren von normalverteilten Zufallszahlen wird in dem Programm «RNORMAL» der Box-Muller Algorithmus verwendet.

4 Die Modellkritik von Mandelbrot

Benoît Mandelbrot, der durch seine Studien über Fraktale einem breiten Publikum bekannt wurde, ist ein heftiger Kritiker der herrschenden Kapitalmarkttheorie. Sein Argument ist einfach[7]: Kursausschläge von mehr als 10σ sind für die Gausssche Glockenkurve extrem selten und haben eine Wahrscheinlichkeit von nur etwa 10^{-23}. Kursausschläge von mehr als 10σ werden an den Börsen aber recht häufig festgestellt. Daher beschreibt das Modell die Wirklichkeit nicht richtig.

Mandelbrot hat seit den 60er Jahren eine grosse Menge an Zeitreihen unterschiedlicher Aktienkursindizes untersucht, um seine Kritik zu belegen und die von ihm vorgeschlagenen Modelländerungen zu testen. Seine Argumente haben jedoch keine grosse Verbreitung erfahren. Dies liegt m.E. daran, dass die stochastischen Prozesse, die er als Ersatz für den Wiener Prozess vorschlägt, über eine unendliche Varianz verfügen und von daher weder der zentrale Grenzwertsatz anwendbar ist noch sonst klassische Statistik betrieben werden kann.

Mandelbrot betrachtet die Klasse der Funktionen exp{-const $|z|^{\alpha}$}. Diese Klasse lässt sich für $\alpha \leq 2$ interpretieren als charakteristische Funktion für eine Reihe von Wahrscheinlichkeitsverteilungen, die man symmetrisch stabile oder symmetrisch L(évy)-stabile Funktionen nennt.[8] Für $\alpha = 2$ erhält man gerade die Normalverteilung, für $\alpha = 1$ die Cauchy-Verteilung und für $1<\alpha<2$ eine Folge weiterer W-Funktionen ohne eigentlichen Mittelwert und mit unendlicher Varianz. Die Funktionen können, bis auf die Ausnahmen Gauss und Cauchy, nicht in geschlossener Form angegeben werden. Ihre Dichte lautet:

$$\frac{1}{\pi}\int_0^x \exp(-u^{\alpha}) \cdot \cos(ux)\, du$$

Mandelbrot kommt durch das Studium von Zeitreihen und durch Interviews von Tradern (Bell Telephone Lab) zu der Überzeugung, dass der Exponent $\alpha = 1.7$ sehr gut das Börsengeschehen modelliert. Diese Charakterisierung ist allerdings keine Universalkonstante, sondern eine empirische Feststellung, die sich im Laufe der Zeit auch verändern kann.

7 B. Mandelbrot, Fractales, Hasard et Finance, Flammarion 1997

8 nach Arbeiten von Paul Lévy, um 1920

Der unten stehende Graph (siehe Abbildung 5) stellt die Dichten von Normal-, Cauchy- und einer L-stabilen Funktion (α = 1.7) dar. Man beachte, dass die Fläche unter der dargestellten Cauchy Funktion nur rund 93% ausmacht, dass also noch 7% Wahrscheinlichkeitsmasse in den Schwänzen jenseits von x = ±10 liegt.

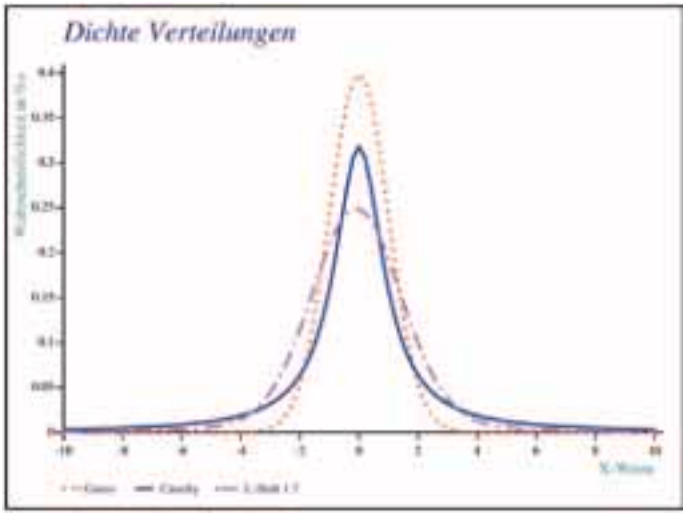


Abbildung 5: Dichten von Normal-, Cauchy- und einer L-stabilen Funktion

4.1 Die Cauchy-Verteilung

Diese Verteilung wurde von Augustin Cauchy im August 1853 vorgestellt[9] . Sie ist definiert durch:

$$f(x) = \frac{1}{\pi(1+x^2)} \qquad ; p.d.f.$$

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \cdot \arctan x \qquad ; i.f.$$

$$x_{ran} = \tan\left(\pi\left(u - \frac{1}{2}\right)\right) \qquad ; ZV$$

Die im Anhang wiedergegebenen APL-Programme sind eine 1:1-Umsetzung dieser Formeln.

Die Verteilung ist dadurch gekennzeichnet, dass sie über «fette Schwänze» verfügt und sich keine endliche Varianz bestimmen lässt. An der Diskussion in der Académie des Sciences in Paris anlässlich ihrer Präsentation wurde die Bedeutung dieser Funktion klar erkannt, zeigte sie doch, dass die Methode der kleinsten Quadrate nicht auf alle Daten – unabhängig vom Verteilungstyp – angewandt werden darf. Auch bezüglich des Erwartungswertes erkannte man, dass die Mittelung die Schätzung nicht verbessert, ja, dass ein Ereignis die Summe der vorherigen Ereignisse auslöschen kann ("*...dass die gleiche Gefahr bei 1.000 Beobachtungen herrscht wie bei einer Einzigen*"). Kurz: der zentrale Grenzwertsatz gilt nicht, also ist die Cauchy Funktion, wie Poisson und Bienaymé folgerten, auch praktisch nicht verwertbar.

Ersetzt man Mittelwert durch Median und Standardabweichung durch die Breite der Verteilung, gemessen am Interquartilbereich, so kann man gleichwohl eine Cauchy-Bewegung analog der Brownschen Bewegung definieren.

Man beachte die Skala der Wahrscheinlichkeiten bei nachstehenden Graphiken!

Bei Abbildung 7 bleiben die Ausschläge ganz nah an dem Median 0 (Skala!). Abbildung 8 gibt neben der ersten Reihe C(540)-1 aus Abbildung 7 zwei deutlich extremere Verläufe wieder. Insgesamt wurden nur 10 Versuche vorgenommen. Die Ergebnisse entsprechen den Charakteristika der Cauchy-Funktion.

Abbildung 7: Skala der Wahrscheinlichkeiten nach Cauchy

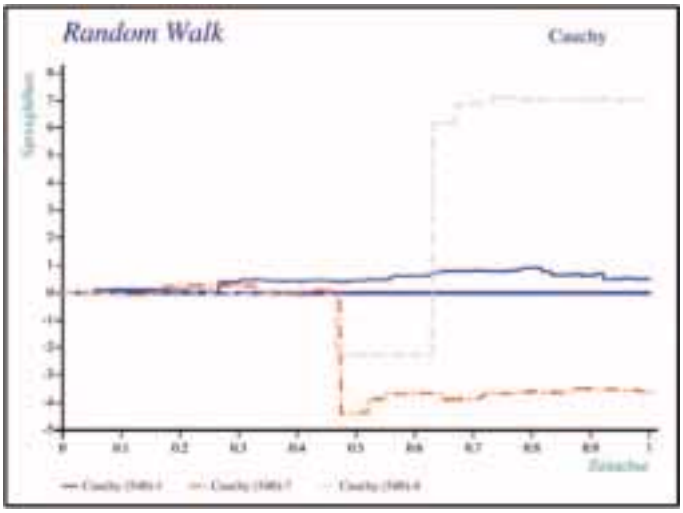


Abbildung 8: Skala der Wahrscheinlichkeiten nach Cauchy

Als Charakteristika der Cauchy Funktion lassen sich festhalten:

- Der «Mittelwert» μ ist identisch verteilt wie das einzelne X_i (Cauchy), die Durchschnittsbildung reduziert nicht die Schwankung, das Gesetz der grossen Zahlen gilt nicht.
- Die Varianz ist unendlich.
- Konzentration: Eine einzelne Zufallsvariable kann – relativ gesehen – das Gros der Summe ausmachen.
- Die Inkremente des Cauchy-Prozesses $\Delta(t,d)=C(t+d) - C(t)$ sind Cauchy verteilt und voneinander unabhängig. $\Delta(t,d)/d$ ist somit unabhängig von t und d.

4.2 Die L-Stabile Verteilung mit α = 1.7

Diese Verteilung soll nach Mandelbrot gegenwärtig den Verlauf des Börsengeschehens bei Aktien am besten widerspiegeln. «Gegenwärtig» heisst, dass Mandelbrot das Muster der Kursbewegungen nicht als historisch invariant ansieht, sondern durchaus als

9 A. Cauchy, Sur les résultats les plus probables, 1853

zeitabhängig. Er konstatiert z.B. eine Änderung der Muster in der Periode 1930 bis 1950.

Auch für die L-Stabile Verteilung ($\alpha = 1.7$) sind die Inkremente voneinander unabhängig und die Varianz ist unendlich. Die Mittelbildung ist nicht ganz obsolet wie bei Cauchy, reduziert aber die Unsicherheit weniger stark als im Gaussschen Fall, nämlich «nur» um $N^{1/1.7}$. Somit eignet sich dieser Prozess ebenfalls als Ersatz für den Wiener Prozess.

Zum Generieren der zugehörigen Zufallsvariablen nutzt man folgende Gleichung[10] aus:

$$\mu + \tau \frac{\sin(\alpha V)}{(\cos(V))^{1/\alpha}} \cdot \left[\frac{\cos((1-\alpha)V)}{W}\right]^{1-1/\alpha}$$

mit

$$V \sim U\left(-\tfrac{\pi}{2}:\tfrac{\pi}{2}\right) \quad U - \textit{gleichverteilt}$$

$$W \sim \textit{Exponentialverteilt}$$

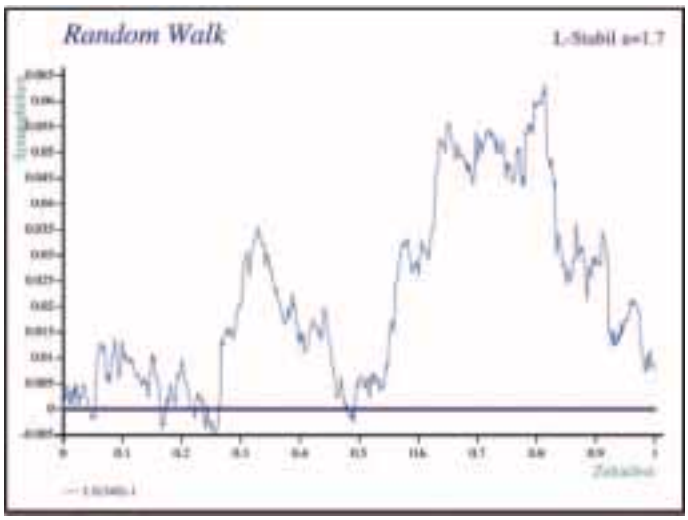


Abbildung 9: L-Stabile Verteilung mit $\alpha = 1.7$

5 Ausklang

Die älteste geschlossene Zeitreihe, die wir kennen ist der Dow-Jones-Index mit Beginndatum 26-V-1896. Einen deutlich „nicht-normalen“ Verlauf weist der japanische Nikkei Index in jüngerer Vergangenheit auf. Die Zeitreihen zu beiden Indizes befinden sich in Abbildung 10 und 11.

Abb. 10: Dow Jones-Index (in US-$) 26.6.1896-24.10.2000

Abb. 11: Nikkei-Index (in Yen) 4.1.1991-24.10.2000

10 B.D. Ripley, Stochastic Simulation, Wiley, 1986

Kontakt

Wolfgang Strasser, Dotzheimer Str. 51, 65197 Wiesbaden, eMail: strasser@gmx.net

Anhang — APL Programme

uniform

```
      ∇ uniform←{                                  ⍝ Uniform Random variates
[1]
[2]     α←0 1
[3]     α[1]+(-/⌽α)×(?⍵⍴¯2+R)÷¯1+R←2*31
[4]     }
      ∇
```

R←RNORMAL

```
      ∇ R←RNORMAL N;V
[1]     ⍝
[2]     ⍝        N Random Normal Deviates by Box-Muller Method
[3]     ⍝
[4]     ⍝
[5]       R←(÷¯1+R)×?(2,⌈N÷2)⍴¯2+R←2*31
[6]       R←N↑(,1 2∘.○R[2;]×○2)×V,V←(¯2×⍟R[1;])*0.5
      ∇
```

W←brown RA

```
      ∇ W←brown RA;a;Xj;t                    ⍝ Brownsche Bewegung
[1]                                          ⍝ RA Anzahl der Schritte N
[2]
[3]     a←u.UNIFORM RA                       ⍝ gleichverteilte ZV generieren
[4]     Xj←(a>0.5)+¯1×a≤0.5                  ⍝ in ¯1 & 1 konzentrieren
[5]     t←(⍳RA)÷RA                           ⍝ Einheit 0 - 1
[6]
[7]     W←0,+\Xj÷RA*0.5                      ⍝ Start 0
      ∇
```

W←brown_N RA

```
      ∇ W←brown_N RA;a;Xj;t                  ⍝ Brownsche Bewegung
[1]                                          ⍝ RA Anzahl der Schritte N
[2]
[3]    Xj←RNORMAL RA                         ⍝ Gauss ZV generieren
[4]    t←(⍳RA)÷RA                            ⍝ Einheit 0 - 1
[5]                                          ⍝ (W(t+d)-W(t))÷d*0.5  ist N(0,d*0.5) verteilt
[6]    W←0,+\Xj÷RA*0.5                       ⍝ Start 0
      ∇
```

W←RW_Cauchy RA

```
      ∇ W←RW_Cauchy RA;a;Xj;t                ⍝ Random Walk Cauchy
[1]                                          ⍝ RA Anzahl der Schritte N
[2]
[3]     Xj←RCauchy RA                        ⍝ gleichverteilte ZV generieren
[4]     t←(⍳RA)÷RA                           ⍝ Einheit 0 - 1
[5]                                          ⍝(W(t+d)-W(t))÷d  ist Cauchy(0) verteilt
[6]     W←0,+\Xj÷RA                          ⍝ Start 0
      ∇
```

W←RW_lstab RA

```
      ∇ W←RW_lstab RA;a;Xj;t                 ⍝ Random Walk L-Stab
[1]                                          ⍝ RA Anzahl der Schritte N
[2]
[3]     Xj←0 1.7 R_LSTABLE RA                ⍝ L-Stab ZV generieren
[4]     t←(⍳RA)÷RA                           ⍝ Einheit 0 - 1
[5]     W←0,+\Xj÷RA                          ⍝ Start 0
      ∇
```

R←{MUE}RCauchy SS

```
      ∇ R←{MUE}RCauchy SS
[1]
[2]    ⍝   Random Draw CAUCHY Fu
[3]    ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[4]    ⍝   L (Levy) Stable Distributions with char. fu. exp{i mue t -|ct|*α}
[5]    ⍝   α=1 Cauchy; α=2 Gauss; 0<α≤2 family; no moments
[6]    ⍝   Moments α<2 - mue=median=loc. para, si=Q3-Q1 interquartil, tau scale para
[7]    ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[8]    ⍝     Cauchy d.f.        :  f(x)=1÷(tau×pi×(1+((x-MUE)÷tau)*2)
[9]    ⍝            i.f.        :  F(x)=0.5+(1÷pi)×arctan ((x-MUE)÷tau)
[10]   ⍝            Ran         :  X=MUE+tau×tan(pi(U-0.5));  U uniform
[11]   ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[12]   ⍝⍝  LA: Parameter MUE; default MUE=0; tau=1
[13]   ⍝⍝  RA: SS No. of random draws
[14]   ⍝⍝   R: XX Random events Xi
[15]
[16]    ⍎(0=⎕NC∇MUE∇)/∇MUE←0'            ⍝ Default {LA} FU RA
[17]
[18]    R←MUE+3○(○1)×(uniform SS)-0.5     ⍝ Xi random Cauchy
      ∇
```

R←MUE_alpha R_LSTABLE SS

```
     ∇ R←MUE_alpha R_LSTABLE SS;MUE;alf;const;V;W
[1]
[2]    ⍝   Random Draw Symmetric Stable Distributions
[3]    ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[4]    ⍝   L (Levy) Stable Distributions with char. fu. exp{i mue t -|ct|*α}
[5]    ⍝                               α=1 Cauchy; α=2 Gauss; 1<α≤2 family
[6]    ⍝                               α=1.7 - Aktienkurse
[7]    ⍝                               c - Skalenparameter:0<c*1÷α = ∆Median-Quartil
[8]    ⍝                                    da keine höheren Momente)
[9]    ⍝   Moments α<2          : mue=Median
[10]   ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[11]   ⍝     V ~ uniform (-pi/2;pi/2)     W ~ expo
[12]   ⍝   Ran:   X  =  ((cos((1-α)×V)÷W)*(1-α)÷α
[13]   ⍝          X  =  MUE+const×(sinαV÷(cos V)*1÷α)×X
[14]   ⍝   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[15]   ⍝⍝  LA: Parameter MUE_α;
[16]   ⍝⍝  RA: SS No. of random draws
[17]   ⍝⍝  R: XX Random events Xi
[18]
[19]   MUE alf←MUE_alpha                        ⍝ seperate
[20]   const←1÷2*1÷alf                          ⍝ c DEFAULT
[21]
[22]   V←((-○0.5),○0.5)uniform SS
[23]   W←-⍟uniform SS                           ⍝ Random Neg. Exponential
[24]
[25]   R←((2○(V×1-alf))÷W)*(1-alf)÷alf
[26]   R←MUE+const×((1○(V×alf))÷(2○V)*1÷alf)×R
     ∇
```

R←BSput RA

```
     ∇ R←BSput RA;K;B;T;si;rf;d1;d2             ⍝ PUT Preis nach Black Scholes
[1]
[2]    ⍝ Preis einer Europäischen Put Option P
[3]    ⍝ das Recht Aktien mit gegenwärtigem Kurs K zum Basispreis B
[4]    ⍝ in T Jahren zu verkaufen. Macht Sinn, wenn K(T)<B+P
[5]
[6]    ⍝  1.  CALL  Preis~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[7]
[8]    ⍝   RA [1             2          3                4       5]
[9]    ⍝       K             B          T                si      rf
[10]   ⍝    Aktienkurs T0  Basispreis  Restlaufzeit J  Vola  RisikoloserZins
[11]   ⍝    current stock  exercise price
[12]
[13]     K B T si rf←RA                          ⍝ separieren
[14]     rf←⍟rf                                  ⍝ ⍟1.03=0.02956 stetiger Zins
[15]
[16]     d1←(⍟K÷B)+T×rf+0.5×si*2  ⋄  d1÷←si×T*0.5
[17]     d2←(⍟K÷B)+T×rf-0.5×si*2  ⋄  d2÷←si×T*0.5
[18]
[19]     C←(K×(u.NIF d1))-B×(*-rf×T)×u.NIF d2    ⍝ Call Preis
[20]
[21]     ⍝ 2. PUT CALL Parität ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[22]
[23]     R←(C+B×*-rf×T)-K                        ⍝ PUT Preis
       ∇
```

Z←NIF

```
     ∇ Z←NIF X;HI
[1]    ⍝
[2]    ⍝      Hastings approximation to the Normal Integral Function,
[3]    ⍝      NBS Handbook, AMS55, nach Anscombe S: 377
[4]    ⍝      Absolute error in Z less than 1E¯7
[5]    ⍝      vgl. Sachs: Angewandte Statistik S: 145
[6]    ⍝
[7]      ⎕PP←16
[8]      HI←319381530 ¯356563782 1781477937 ¯1821255978
[9]      HI←HI,1330274429
[10]     Z←0.5+(×X)×0.5-(*X×X÷¯2)×((÷1+0.2316419×|X)∘.*⍳5)+.×HI×3.989422804×10*¯10
[11]     ⎕PP←10
     ∇
```

IBM

Today's APL2

APL2

Ausgabe Nr. 7 Auszugsweise Übersetzung von „News from APL Products and Services at IBM Corporation" Mai 2002

Neu! Workstation APL2 Version 2.0

Neues Produkt, neue Plattform, neue Fähigkeiten

IBM hat das neue Workstation APL2 Version 2.0 zum Vertrieb freigegeben. Es enthält die Unterstützung für die Betriebssysteme AIX, Linux, Sun/Solaris und Windows auf einer einzigen CD.

Das neue Linux APL2 läuft auf PC-kompatiblen Maschinen und hat dieselben Fähigkeiten wie APL2 für AIX und Sun Solaris.

Neue Fähigkeiten kennzeichnen Version 2.0: Unterstützung für Namespaces, Schnittstelle zur TCL-Befehlssprache, Online-Dokumentation im PDF-Format, APL2 Runtime Library zur kostenlosen Verteilung von Anwendungen und eine neue AP145-Tabellensteuerung (grid control) für Windows-Systeme.

Der Listenpreis für Workstation APL2 V2.0 beträgt 1769 €. Mengenrabatte und reduzierte Preise für die Aufrüstung von APL2 V1 Produkten (AIX, OS/2, Sun Solaris, Windows) sind über das IBM Passport Advantage Programm verfügbar.

APL2 Namespaces

APL2 Namespaces sind gekapselte Anwendungen außerhalb des aktiven APL2-Arbeitsbereiches. Namen können mit Objekten in Namespaces assoziiert und als Objekte im Arbeitsbereich benutzt werden.

Namespaces bieten viele Vorteile: Namenskonflikte werden reduziert durch Isolierung von Anwendungen in ihrer eigenen Namensumgebung. Vorhandener Code kann leichter wieder benutzt werden durch Speicherung von Anwendungen in Namespace-Dateien außerhalb des Arbeitsbereiches. Anwendungs-Interna werden geschützt durch Begrenzung des Zugriffs von außen auf nur wenige Objekte.

Die Namespace-Unterstützung des neuen Produkts ist voll kompatibel mit der schon lange im APL2/390 vorhandenen. Damit können jetzt Namespace-Anwendungen entwickelt werden, die auf Workstations und Großrechnern laufen können.

APL2 für Ausbildung

Das „IBM Scholars Program" ist ein kostenloses Angebot an höhere Ausbildungsinstitute. Es erleichtert den in Lehre und Forschung Tätigen, Produkte und Technologien, Ausbildungsressourcen und andere Informationen von IBM zu erhalten.

APL2 Runtime Library

Diese neue Ausführungsbibliothek ist eine Untermenge des APL2V2-Systems, mit der Anwendungen kostenlos weitergegeben werden können, ohne dass die empfangenden Benutzer APL2-Lizenzen beschaffen müssen.

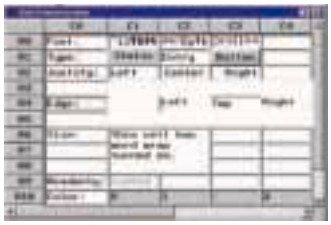
Der mit APL2 Runtime Library gelieferte Interpretierer besitzt alle Möglichkeiten des normalen APL2-Interpretierers, nur Sitzungsmanager und Systembefehle werden nicht unterstützt. Um Runtime-Anwendungen, die auch in Form von Namespaces erzeugt werden können, zu starten, können Aufrufparameter oder APL2-Eingabedateien benutzt werden.

Eine Version der APL2 Runtime Library wird mit jeder von Workstation APL2 Version 2.0 unterstützten Plattform mitgeliefert. Im mitgelieferten englischen APL2 User's Guide ist dargestellt, wie man APL2-Runtime-Anwendungen schreiben kann.

Das Allerbeste: Das „IBM Scholars Program" enthält jetzt APL2!

Für APL2 im „IBM Scholars Program" nähere Informationen bei : Klaus Drebenstedt, IBM Informationssysteme GmbH, Hauptverwaltung Stuttgart, eMail: KWD@de.ibm.com.

AP145 Tabellensteuerung



Diese neue Möglichkeit im AP145 kann Daten nach Art einer Tabellenkalkulation darstellen. Einzelne Zellen können Eingabe- oder Lesefelder, Schaltknöpfe oder statische Textbereiche sein, wahlweise mit Zeilenumbruch, deren Größe, Farben, Zeichensätze, Ausrichtung und Kantenform einzeln festgelegt werden können.

TCL und TK

TCL ist eine populäre, für viele Betriebssysteme verfügbare Scripting-Sprache. Workstation APL2 V2.0 fügt zu TCL eine neue externe Schnittstellenfunktion hinzu. Die APL2-Funktion TCL schickt beliebige TCL-Befehle an den TCL - Interpreter und konvertiert automatisch die Datentypen zwischen APL2 und TCL.

TK ist ein TCL-Erweiterungspaket zum Aufbau grafischer Benutzerführungen (GUI). Damit können plattformunabhängige GUI-Anwendungen entwickelt werden, die alle Möglichkeiten von APL2 zur Datenmanipulation nutzen können.

Weitere Informationen zu APL2 direkt vom IBM Silicon Valley Labor:
E-mail: apl2@vnet.ibm.com http://ibm.com/software/ad/apl

Workstation APL2 for Multiplatforms Version 2.0

Workstation APL2 for Multiplatforms Version 2.0, entwickelt vom APL Products and Services Team im IBM Silicon Valley Labor in San Jose, Kalifornien, liefert voll funktionierende APL2-Umgebungen für die Betriebssysteme AIX, Linux, Sun Solaris und Windows.

Workstation APL2 for Multiplatforms Version 2.0, entwickelt vom APL Products and Services Team im IBM Silicon Valley Labor in San Jose, Kalifornien, liefert voll funktionierende APL2-Umgebungen für die Betriebssysteme AIX, Linux, Sun Solaris und Windows.

APL2 ist eine Programmiersprache, die sowohl für Anwendungsentwickler als auch für interaktive Endbenutzer geeignet ist. Profis in der Datenverarbeitung sehen in APL2 ein hochproduktives Werkzeug zur Entwicklung von kommerziellen und wissenschaftlichen Anwendungen. APL2 wird ebenso als außerordentlich effektiv angesehen in so unterschiedlichen Bereichen wie „Was-wäre-wenn"-Modellierung, experimentelle Programmierung, interaktives (Intensiv-)Rechnen, Entscheidungsunterstützung und Datenanalyse.

Workstation APL2 Version 2.0 ist kompatibel mit den APL2 Version 2 Produkten, die auf Großrechnern IBM zSeries mit CMS und TSO laufen, sowie mit den APL2 Version 1 Produkten für AIX, OS/2, Sun Solaris und Windows. Alle APL2-Systeme erlauben transparentes Im- und Exportieren von Programmen und Daten and sichern einen hohen Grad von Anwendungsportabilität.

Workstation APL2 Version 2.0 kombiniert alle Vorteile und Einrichtungen der bisherigen APL2 Version 1 Produkte für AIX, Solaris und Windows, die es ersetzt. Darüber hinaus bringt V2.0 wichtige Neuerungen wie:

- Unterstützung von Linux auf PC-kompatiblen Systemen
- Namespaces zur Einkapselung und Weitergabe von Anwendungen
- Schnittstelle für die TCL/TK-Befehlssprache
- Online-Dokumentation im PDF-Format
- APL2 Runtime Library zur (kostenlosen) Verteilung von APL2-Anwendungen

Das Wichtigste in Kürze

Entwicklungsumgebung:

Interaktives Programmieren wird durch einen GUI-Sitzungsmanager unterstützt. Unter Windows sind die Editoren für Programme, Daten und Dialoge integriert mit dem Sitzungsmanager.

Grafische Benutzerführung (GUI):

Unter Windows liefert APL2 Werkzeuge zum Aufbau von GUI-Anwen-dungen. Version 2.0 fügt eine ausgefeilte Tabellenanzeige hinzu.

Große Arbeitsbereiche

APL2 nutzt die 32-bit-Architektur von Windows, die Anwendungen ermöglicht, in Arbeitsbereichen bis nahe an 2 GB zu arbeiten.

(Englische) Online Dokumentation

„APL2 Users' Guide", „APL2 Language Reference Manual", „ APL2 Language Summary", „APL2 Programming: Using SQL" und „GRAPHPAK User's Guide" werden online in PDF-Format mitgeliefert.

Unterstützung von Landessprachen

APL2 akzeptiert Befehle und gibt Nachrichten in verschiedenen Landessprachen aus. Daten in Unicode werden von der TCL-Schnittstelle und vom HTTP-Server unterstützt.

Runtime Library

Mit dieser Untermenge des APL2-Systems können Anwendungen kostenlos weitergegeben werden, ohne dass die Empfänger APL2-Lizenzen beschaffen müssen.

Internet-Werkzeuge

Der Arbeitsbereich NETTOOLS und unterstützende Dienstprogramme liefern einen HTTP 1.1 Webserver und Client, die Datentransfer und „remote procedure calls" unabhängig vom APL-Dialekt ermöglichen.

Verteilte Verarbeitung

APL2 bietet die Möglichkeit, Daten und Verarbeitung auf mehrere, über TCP/IP verbundene APL2-Systemen zu verteilen. Die APL2-Systeme unter TSO, CMS, AIX, Linux, Sun Solaris und Windows haben alle die folgenden Einrichtungen:

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

Namespaces

Namespaces ermöglichen das Einkapseln von Arbeitsbereichen und damit reduzierte Namenskonflikte bei der Wiederbenutzung. Da Namespaces auf Workstations unterstützt werden, können Anwendungen entwickelt und getestet werden, die als Namespaces auf Großrechnern laufen sollen.

Dateien als Variable

Dateien können wie APL2-Variable mit APL2-Elementarfunktionen und -Operatoren verarbeitet werden.

Objektbibliotheken

APL2-Objekte können über ihre Namen gespeichert werden. Objektbibliotheken sind portabel zwischen allen APL2-Systemen.

IBM®
Workstation APL2
for Multiplatforms Version 2.0
IBM Programm-Produkt-Nr. 5724-B74, Teile-Nr. 45P7514

Universelle Grafikunterstützung

Der Grafikprozessor erzeugt Grafiken, Text und Bilder von hoher Qualität mit Hilfe einer einfachen Befehlssyntax. Der Arbeitsbereich GRAPHPAK ist kompatibel mit denen aller anderen APL2-Systeme.

Datenbankzugriff

Der Hilfsprozessor AP127 bildet die Schnittstelle zu IBM DB2, AP227 zu Datenbanksystemen mit ODBC-Unterstützung. In beiden Fällen wird die SQL-Sprache benutzt.

TCL-Schnittstelle

Die TCL-Befehlssprache liefert eine große Vielfalt von Routinen und Werkzeugen, z.B. zur Untersuchung von Zeichenketten, zur Datei-Ein/-Ausgabe und zum Aufbau grafischer Benutzerführungen (GUI). Diese Einrichtungen können über die TCL-Schnittstelle von APL2 benutzt werden.

Performance-Monitor

Der Arbeitsbereich TIME erlaubt die Analyse von APL2-Programmen zum Auffinden von Performance-Engpässen.

Arbeitsbereiche zur Anwendungsentwicklung

Beispiele und Dienstprogramme in mitgelieferten Arbeitsbereichen können direkt benutzt oder abgewandelt werden.

Werkzeuge zum Schreiben von Hilfsprozessoren

Musterprozessoren, C header files und eine Serverschnittstelle erlauben eine leichte Implementierung von Spezial- und Client-Serverprozessoren.

Aufrufe von Programmen in anderen Sprachen

In anderen Sprachen geschriebene Programme können in der normalen APL2-Syntax aufgerufen werden.

System Services

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

Als assoziierte Prozessoren werden geliefert:

P10 REXX-Prozessor: erlaubt die Benutzung von REXX-Funktionen wie APL2-Funktionen, Manipulation von REXX-Programmen wie APL2-Strukturen und leichten Zugriff auf Betriebssystemdateien.

P11 Aufruf externer Routinen: ein APL2-Programm kann damit Routinen außerhalb des APL2-Arbeitsberei-ches aufrufen, die in einem APL2-Namespace oder in einer anderen Sprache sind. Verschiedene P11-Routi-nen sowie Dokumentation und Musterdateien zur Entwicklung eigener externer Routinen werden mitgeliefert.

P12 Dateien als Variable: Dateien (Textdateien des Betriebssystems und solche im AP210-Format) können wie APL2-Variable mit APL2-Elementarfunktionen und -Operatoren verarbeitet werden.

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 die Netz-Kommunikation sicherstellt.

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

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

AP144 X Window API (nur für Unix-Systeme): ist die Schnittstelle zu einer großen Zahl von Xwindow-Systemaufrufen (Xlib). Arbeitsbereiche für Cover-Funktionen und Demos sind beigefügt.

AP145 GUI Serviceprocessor: ermöglicht den Zugriff auf Services der grafischen Benutzerführung. Arbeitsbereiche für Cover-Funktionen und Demos sind beigefügt.

AP207 Universeller Grafikprozessor: erzeugt Grafiken, Text und Bilder von hoher Qualität mit Hilfe einer einfachen Befehlssyntax. Arbeitsbereiche für Demos und GRAPHPAK sind beigefügt.

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

AP211 APL2 Objektbibliotheks-Prozessor: zum Schreiben und Lesen von APL2-Strukturen über ihre Namen. Die damit erzeugten Dateien sind portabel auf alle APL2-Plattformen (die von APL2/DOS sind lesbar).

AP227 ODBC-Prozessor: zum Zugriff über SQLauf alle DB-Systeme und Anwendungen, die das ODBC-Protokoll unterstützen.

AP488 GPIB-Prozessor: zum Zugriff auf externe Geräte, die über den General Purpose Interface Bus (GPIB) und National Instruments GPIB Treiber angeschlossen sind. (nur für Windows Systeme).

Technische Informationen

Hardware-Voraussetzungen:

- Mindestens 25 MB freier Plattenplatz
- Mindestens 15 MB Hauptspeicher über den Bedarf des Betriebssytems hinaus

Software-Voraussetzungen:

Workstation APL2 Version 2.0 hat die folgenden Software-Mindestvoraussetzungen:

- AIX 3.2.5 mit Motif 1.2 und X Window System X11R5
- Eine Linux-Distribution für PC-kompatible Maschinen mit Motif 2.1 und X Window System X11R6

- Solaris 2.3 mit Motif 1.3 und X Window System X11R5
- Windows 98, Windows ME, Windows NT, Windows 2000, oder Windows XP

Einige Einrichtungen von Workstation APL2 haben zusätzliche Mindestvoraussetzungen:

- Adobe Acrobat Reader ist erforderlich zur Darstellung der Online-Dokumentationen.
- IBM DB2 Universal Database, Version 5.0 für AP 127
- Ein ODBC Driver Manager, der ODBC 3.0 Application Programming Interfaces unterstützt, für AP 227.
- TCP/IP für AP 119 und systemübergreifende gemeinsame Variable.
- IBM Object REXX zum Aufruf von REXX-Routinen mit Prozessor 10
- TCL und TK 8.1 für die externe TCL-Funktion.

Englische Dokumentationen:

Workstation APL2 Version 2.0 enthält die folgenden englischen Dokumentationen im PDF-Format:

- APL2 User's Guide (SC18-7021)
- APL2 Programming: Language Reference (SH21-1061)
- APL2 Programming: Using SQL (SH21-1057)
- APL2 GRAPHPAK: User's Guide and Reference (SH21-1074)
- APL2 Language Summary (SX26-3851)

Außerdem enthält Workstation APL2 Version 2.0:

- APL2 Tastatur-Aufkleber SC33-0604 (Diese Tastatur-Aufkleber sind nicht enthalten, wenn das Produkt heruntergeladen wird.)

Kompatibilität:

Workstation APL2 V2.0 ist kompatibel mit den APL2V1-Produkten für AIX, OS/2, Sun/Solaris und Windows. Mit diesen Produkten geschriebene Anwendungen laufen auch auf Workstation APL2 V2.0, das aber ein neues Format für die Arbeitsbereiche benutzt. D.h., mit V2.0 gespeicherte Arbeitsbereiche können nicht in V1.0-Systemen geladen, sondern nur durch)OUT und)IN zurück migriert werden.

Für weitere Informationen[1] können Sie die APL-Websites von DPC oder von IBM aufsuchen:

- http://www.apl-online.de
- http://www.ibm.com/software/ad/apl

1 Workstation APL2 Version 2.0 kann bei DPC nicht nur neu bestellt, sondern auch von einem der Vorgängerprodukte APL2 Version 1.0 für AIX, OS/2, Solaris oder Windows über das IBM Vertriebsprogramm „Passport Advantage" günstig aufgerüstet werden. Diese Veröffentlichung von DPC ist eine auf deutsche bzw. europäische Verhältnisse angepasste Übertragung des IBM Dokuments „Workstation APL2 for Multiplatforms Version 2.0 Technical Brief" vom Mai 2002 und kann nur allgemeine Hinweise geben. IBM, APL2, OS/2, AIX, und DB2 sind Warenzeichen der IBM Corporation. Sun and Solaris sind Warenzeichen von Sun Microsystems, Inc. Windows and Windows NT sind Warenzeichen von Microsoft Corporation. Linux ist ein Warenzeichen von Linus Torvalds. Adobe und Acrobat sind Warenzeichen von Adobe Systems, Inc. Motif ist ein Warenzeichen von The Open Group.
Einzelheiten dazu entnehmen Sie bitte dem DPC-Blatt „Bestellinformationen für IBM Workstation APL2" oder der Web-Seite gleichen Namens unter der URL von DPC www.apl-online.de.

Kontakt

Dittrich & Partner Consulting GmbH
Kieler Str.17, 42697 Solingen, Telefon: 0212-26066-0, Fax: 0212-26066-66, eMail:info@dpc.de

IBM „APL Products and Services"
Dept. H36A/F40, IBM Silicon Valley Laboratory, 555 Bailey Avenue, San Jose, CA 95141, USA
Telefon: 001-408-463-APL2 (001-408-463-2752), Fax: 001-408-463-4488, eMail:apl2@vnet.ibm.com,
News group: news://news.software.ibm.com/ibm.software.apl

http://www.apl-germany.de

APL Germany e.V.

- Homepage
- Wir über uns
- Was ist APL?
- Wer bietet APL an?
- Veranstaltungen
- Literaturhinweise
- APL-Journal
- APL-Nachrichten
- APL-Links

NEW! Herbst-Tagung

Webmaster: webmaster@hoesle-kienzlen.de

Der *APL Germany e.V.* ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache *APL*, sowie darüberhinausgehend die Verbreitung des Verständnisses der Mensch - Maschine - Kommunikation zu fördern.

Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet *APL Germany e.V.* den adäquaten organisatorischen Rahmen.

Zu diesem Zweck veranstaltet der *APL Germany e.V.* beispielsweise Vorträge, Diskussionen und APL-Kurse, stellt Kontakte zu Computerfirmen her, die APL anbieten, und gibt das Mitteilungsblatt „APL-Journal" heraus.

Um die Kommunikation der Mitglieder untereinander zu erleichtern, hat APL Germany einen Nachrichten-Server eingerichtet. Bei diesem Server kann sich jeder, der an APL interessiert ist einschreiben. Eine Mitgliedschaft bei *APL Germany e.V.* ist nicht erforderlich (aber natürlich gerne gesehen).

Kontaktadresse:
APL Germany e.V.
Rheinstr. 23

D-69190 Walldorf

info@apl-germany.de

Allgemeine Informationen

(Stand Frühjahr 2002)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

Vorstand

Vorsitzender:

Dieter Lattermann
Rheinstraße 23
69190 Walldorf
Tel. (06227) 63469
eMail: 100332.1461@compuserve.com

Schriftführer

Conrad Hoesle-Kienzlen
CHK IT-Consulting
Augustenstr. 57
70178 Stuttgart
Tel. 0172 722 04 48
Fax 0711 674 38 53
eMail: chk@hoesle-kienzlen.de

2. Vorsitzender:

Martin Barghoorn
(Technische Universität Berlin)
Lückhoffstr. 8
14129 Berlin
Tel. (030) 314 24392
Fax: (030) 314 25901
eMail: martin@barghoorn.de
http://stat.cs.tu-berlin.de/~barg/

Schatzmeister

Jürgen Beckmann
Mohnäckerstr. 1
68259 Mannheim
eMail: JBecki@compuserve.com

Beitragssätze

Ordentliche Mitglieder:

Natürliche Personen	31,00 Euro	p.a.
Studenten / Schüler	10,00 Euro	p.a.

Außerordentliche Mitglieder:

Juristische / natürliche Personen	511,00 Euro	p.a.

Bankverbindung:

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

Hinweis:

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

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

Einzugsermächtigung

Mitglieds-Nr.: _____

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: _____