inforum

INFOrmationsforum des Rechenzentrums der Universität Münster

Jahrgang 15, Nr. 2 – April 1991 ISSN 0931–4008

Inhaltsverzeichnis

Impressum infoforum

ISSN 0931-4008

Redaktion: W. Bosse (Tel. 83-2461)
St. Ost (Tel. 83-2681)
H. Pudlatz (Tel. 83-2472)
E. Sturm (Tel. 83-2609)

Satz: A. Ibach
S. Arnold
Satzsystem: LaTeX
Druck: M. Näther
A. Wolters

Universitätsrechenzentrum
Einsteinstraße 60
4400 Münster

Auflage dieser Ausgabe: 1000

Redaktionsschluß der nächsten Ausgabe: 15.6.1991

Editorial

von

E. Sturm

Auf *einen* Artikel warten wir, die Redakteure des infoforum, jetzt schon seit über zwei Jahren – also ungefähr 8 infoforum-Ausgaben. Er steht deshalb auch ganz am Anfang dieses Heftes: Endlich gibt es etwas Definitives zu berichten „Zum Stand der Beschaffungsmaßnahme EDV-Gesamtversorgung".

Ein zukunftsträchtiges Thema ist in dieser Ausgabe mit zwei Artikeln vertreten: Grafik-Dateien.

Bisher schon konnte man auf dem Großrechner Grafik in DCF- und TEX-Dokumente einbinden, im Falle TEX aber nur in einer Weise, die diesem Satzsystem nicht gerecht wurde: Text allein konnte man zwar auch auf PC-Druckern ausgeben, Text mit Grafik aber nur auf dem zentralen P400. Die neue, ebenfalls selbst entwickelte, Software bringt hier nun die für TEX so typische Geräteunabhängigkeit auch bei Bildern!

Der andere Grafik-Artikel beschreibt die Möglichkeit, Bilder aus verschiedenen Quellen, so auch von einem Farb-Scanner, auf dem zentralen Thermotransfer-Drucker ausgeben zu können. Wer sich die Mühe macht, kann auf diese Weise z.B. einen Farbkopierer simulieren (mit ungefähr 2 DM pro Bild, abgesehen von der nicht unbeträchtlichen Rechenzeit). Nicht zu vergessen die Fähigkeit des COLPLOT-Programms, Beschriftungen vornehmen zu können!

Bei beiden Artikeln wird ein Problem sichtbar, das sich wohl prinzipiell nicht lösen läßt, um dessen Linderung sich das Rechenzentrum aber bemüht: Es gibt eine schier unübersehbare Zahl von grafischen Dateiformaten, wie die Kürzel GIF, MSP, TIFF, HPGL, PCL, EPS, WPG, IGES, DXB, DXF oder PCX zeigen. Dabei ist in dieser Liste noch nicht einmal das, wie ich meine (und PostScript-Anhänger mögen mir verzeihen), wichtigste Dateiformat, CGM, enthalten. Das sogenannte Computer Graphics Metafile ist nämlich die einzige international anerkannte Norm auf diesem Gebiet. Es gibt zwar erst wenige PC-Programme, die dieses Format „kennen", dennoch sehe ich hier die einzige Möglichkeit, dem Wirrwarr zu entkommen.

Das Rechenzentrum wird also in Zukunft (wenn erst ein neuer Zentralrechner da ist) alles daransetzen, über CGM einen Austausch von Bildern zu ermöglichen – z.B. zur Ausgabe von auf dem PC erzeugten Bildern auf hochwertigen zentralen Plottern. Wenn Sie also den Kauf von grafikfähiger Software für den PC planen, CGM ist ein Argument!

RUM-Aktuell

Zum Stand der Beschaffungsmaßnahme „DV-Gesamtversorgung“

von

W. Held

Nach einem anstrengenden Marsch durch den langen Tunnel ist nunmehr ein heller Lichtschein zu erkennen, der wohl vom Ende des Tunnels herrührt.

Im Februar 1991 hat die Deutsche Forschungsgemeinschaft ein positives Votum zur DV-Beschaffung in Münster abgegeben. Zusammen mit den Fachbereichen sind daraufhin die Konfigurationen noch einmal aktualisiert worden, als Vorbereitung einer Ausschreibung. Bei günstigem Verlauf werden in diesem Sommersemester die ersten Verträge mit den Lieferanten abgeschlossen werden können.

Im Rahmen dieser DV-Gesamtversorgung sind folgende Maßnahmen vorgesehen:

1. Ausstattung fast aller Fachbereiche mit Arbeitsplatzrechnern, Workstations und den dazu benötigten Servern.
2. Ausstattung der Fachbereiche Physik und Chemie mit Servern und leistungsfähigen Workstations. Zu einem gewissen Anteil sollten dabei VMS-Systeme beschafft werden.
3. Ersatz der zentralen Anlage im Universitätsrechenzentrum durch ein leistungsfähiges IBM-System (oder kompatibel) mit Vektorzusätzen.

Zu den Rechnern gehören natürlich periphere Komponenten und vielfältige Software-Produkte.

In dem DV-Gesamtsystem werden u.a. Window-artige Oberflächen, Network File System (NFS) mit Remote Procedure Call (RPC) zur Verfügung stehen.

Damit können auch Daten, wenn sie nicht auf den eigenen Geräten lagern, leicht genutzt werden; damit kann Software, die zentral vorgehalten wird, zur Ausführung in das eigene Gerät geladen werden. So können z. B. vom Programm eines Rechners Prozeduren in anderen Rechnern aufgerufen werden, die ihre Ergebnisse in das eigene Programm zurückliefern.

Dies wird vielfältige neue Verarbeitungsmöglichkeiten schaffen. Nicht jedes Produkt, das man vorübergehend benötigt, muß sogleich gekauft werden, wenn man es auf einem anderen Gerät nutzen kann.

Vektorisiertes Rechnen wird möglich werden. In begrenztem Umfang wird man über das hier Beschriebene hinaus an paralleles Rechnen in lose gekoppelten Systemen (über das lokale Netz) denken können.

Kurz: Wenn hoffentlich am Ende dieses Jahres die meisten Ressourcen beschafft sein werden, finden Forschung und Lehre einige Jahre lang gute DV-Arbeitsbedingungen vor.

Das Universitätsrechenzentrum ist aufgefordert, seine Unterstützung für die nicht im Rechenzentrum stehenden Ressourcen über das bisherige hinaus noch zu verstärken. Der Organisationsplan wird z.Z. entsprechend angepaßt. Insbesondere wird die Aufteilung der Zuständigkeiten zwischen Fachbereichen und Universitätsrechenzentrum festzulegen sein.

Kermit

von

G. Richter

Das bekannte Public-Domain-Programm KERMIT für Terminalemulation und Dateitransfer ist ab sofort im Dispatch des Universitätsrechenzentrums und über das Software-Verteilungssystem VAMP erhältlich.

Sie erhalten neben dem eigentlichen Programm und zugehöriger Dokumentation eine Benutzeranleitung für den Einsatz im konventionellen Netz der Universität (V.24-Leitungen), insbesondere für den Zugang zum CMS und zum Wissenschaftsnetz WIN über den Gateway-Rechner RUMSWTCH.

Vorzugsweise soll dieses Programm anstelle von MT, GVT und FTERM eingesetzt werden, vorausgesetzt, es sind keine RUM/GKS-Anwendungen betroffen. Eine etwas umfangreichere Einsatzbeschreibung erhalten Sie auf Anfrage beim Universitätsrechenzentrum durch Herrn Speer, der Sie auch weiter beraten kann (Tel. 2679).

S-PLUS

von

Prof. Dr. W. Köpcke

Institut für Med. Informatik und Biomathematik

Das System S-Plus von der Gesellschaft für statistische Informationssysteme in Dortmund ist speziell für Unix-Betriebssysteme entworfen worden, läuft aber inzwischen auch unter MS-DOS.

S-PLUS besitzt eine Schnittstelle zu X-Windows, die dem Anwender eine komfortable Benutzeroberfläche bereitstellt. Der Dialog mit S-PLUS findet über Kommandos statt. Das Design der Sprache ist an die Programmiersprache C angelehnt, doch erlaubt S-PLUS auch höhere Datenstrukturen, mit denen sich komplexe Ausdrücke einfach handhaben lassen. Zudem sind die Befehle sehr mächtig, so daß der Benutzer sich bei der Arbeit fast ausschließlich auf seine inhaltlichen Ziele konzentrieren kann. Auf diese Weise ist eine problemorientierte Arbeit realisiert. Zudem sind Unix-Tools wie *sed*, *grep* und *awk* unter S-Plus voll verfügbar.

Für die Analyse großer Datenmengen ist eine grafische Darstellung oft aussagekräftiger als eine lange Tabelle mit numerischen Werten. Hier zeigt sich die große Stärke von S-Plus, das vielfältige Präsentationsformen bis hin zu 3-D-Grafiken beherrscht. Um eventuell vorhandene Beziehungen zwischen den Werten einer umfangreichen Menge leichter ausmachen zu können, existiert die Darstellungsform *brush & spin*. Sie bildet die Elemente einer Menge als Punktwolke im Raum ab, die dann beliebig drehbar ist. Häufungspunkte und Ausreißer sind auf diese Weise leicht zu ermitteln und erlauben eine effektive Planung der weitergehenden Analyseschritte.

Das System stellt Schnittstellen zu C und FORTRAN zur Verfügung, so daß der Anwender auch eigene Befehle entwickeln kann, die dann in das System eingebunden werden und fest dazugehören. Mit über 500 Funktionen im Basispaket sollte das System eigentlich alle statistischen Ansprüche zufriedenstellen. Doch liegen die Stärken von S-PLUS eher bei den Prozeduren zur erforschenden Datenanalyse.

Der Umfang von S-Plus deutet darauf hin, daß der potentielle Anwender eher im Bereich gesehen wird, wo große Datenmengen anfallen und auszuwerten sind. Als mächtiges Hilfsmittel für Analyse und Grafik stellt es somit eine Ergänzung zu Systemen dar, die bei den Methoden der traditionellen Statistik zu Hause sind.

Eine dreitägige Einführung in S-Plus findet vom 15.–17. Juli im Seminarraum des Instituts für Medizinische Informatik und Biomathematik, Domagkstr. 9, in der Zeit von 14 bis 17 Uhr statt. Anfragen richten Sie bitte an mich (Tel. 5262).

Konvertierung von FORTRAN-Dialekten

von

R. Mersch

Dieser Artikel ist denen gewidmet, die es gewohnt sind, nicht standardgemäß zu programmieren.

Aufgrund der weiten Verbreitung der Programmiersprache FORTRAN im wissenschaftlichen Bereich kommt es immer wieder vor, daß FORTRAN-Programme auf anderen Rechnern als denjenigen, für die sie ursprünglich entwickelt wurden, eingesetzt werden sollen. Leider ist es so, daß die verschiedenen FORTRAN-Compiler durchaus ihre Eigenheiten haben, so daß ein beispielsweise für eine VAX geschriebenes FORTRAN-Quell-Programm vom IBM-FORTRAN-Compiler nicht unbedingt verstanden wird, z. B. dann, wenn man nicht standardkonform programmiert hat. In diesem Fall muß das Quell-Programm konvertiert werden.

Auf unseren VM-Systemen steht jetzt das IBM FORTRAN Translation Tool zur Verfügung, das bei der Konvertierung der folgenden FORTRAN-Dialekte nach VS FORTRAN behilflich ist:

- CDC FORTRAN Versionen 2.3, 4, 5
- DEC PDP-11 FORTRAN IV und FORTRAN IV PLUS
- DEC TOPS 10/20 FORTRAN IV (FORTRAN 10, FORTRAN 20)
- DEC VAX-11 FORTRAN Version 3.0
- DEC VAX FORTRAN Version 4.0

Das FORTRAN Translation Tool erzeugt ein VS-FORTRAN-Quellprogramm auf der Sprachebene FORTRAN 77. Seine Anwendung ist als erster Schritt der Konvertierung zu verstehen; u. U. müssen manuelle Eingriffe folgen.

Folgendes ist zu tun, um ein in einem der angegebenen Dialekte geschriebenes FORTRAN-Programm – es stehe in der Datei `FREMD FORTRAN` – zu konvertieren:

An den Anfang der Datei `FREMD FORTRAN` ist der folgende Text (ab Spalte 1) zu setzen, der angibt, um welchen Dialekt es sich handelt.

```
%PROCESS SOURCE(xxx)
```

Hier steht `xxx` für `CDC`, `PDP`, `PLUS`, `TOPS` oder `VAX`. Mit dem Kommando

```
ENVIRON TRANFORT
```

wird das FORTRAN Translation Tool in Zugriff genommen, und mit dem Kommando

```
XLAT FREMD
```

die Konvertierung gestartet. Das Ergebnis steht anschließend in der Datei `FREMD NEWFORT`, und die Datei `FREMD TRANPRNT` enthält ein Protokoll der Konvertierung. Letztere gibt insbesondere Hinweise auf Stellen, die möglicherweise noch manueller Eingriffe bedürfen.

Angemerkt werden sollte, dass es eine Reihe weiterer Optionen für die `%PROCESS`-Anweisung gibt, mit denen die Konvertierung gesteuert werden kann. Ferner existiert eine Datei namens `TRANDATA FTO5F001`, die den zu konvertierenden Dialekt beschreibt (reservierte Wörter, Schlüsselwörter, etc.), und die u. U. angepaßt werden muß. Näheres hierzu findet sich im Handbuch, das in der allgemeinen DV-Beratung (Programmierberatung) eingesehen werden kann.

Personalia

Seit dem 4.3.1991 ist Frau E. Neumann im Geschäftszimmer des Universitätsrechenzentrums tätig. Sie hat die Aufgaben von Frau I. Lampkowski übernommen, die auf eigenen Wunsch beurlaubt ist.

Bei den studentischen Mitarbeitern haben sich folgende Veränderungen ergeben:

Ausgeschieden sind Frau S. Klessinger zum 31.1.1991 sowie die Herren H. Brinkmann zum 31.1.1991, St. Arntzen und A. Kulka zum 28.2.1991 und G. Hetrodt zum 31.3.1991. Herrn Hetrodt gratulieren wir zum erfolgreichen Diplom in Mathematik.

Am 1.3.1991 hat Herr J. Urbainczyk seine Tätigkeit aufgenommen und am 2.4.1991 Herr R. Sibbel.

Neues im VAMP

von

H. Pudlatz

Das „Verteilungssystem Allgemeiner Mikrocomputer-Programme“ (VAMP) erlaubt es, PC-Software im Rechnernetz zu kopieren. Hierbei ist für die Kategorien Lizenz-Software und Public-Domain-Software (PD) eine unterschiedliche Verfahrensweise vorgesehen.

Falls Sie **PD-Software** kopieren wollen, beantworten Sie die Frage nach der Benutzerkennung mit der RETURN-Taste und kommen dann automatisch in den PD-Bereich. Hier gibt es inzwischen die neue Version 2.8 des früher im emTEX-Paket enthaltenen Produktes TEXCAD. Sie ist jetzt gesondert abrufbar (das emTEX-Paket enthält noch die alte Version). Außer einer Korrektur bei der Darstellung von Vektoren und einer verbesserten Snap-Funktion enthält die neue Version die Möglichkeit, Bezier-Kurven zu zeichnen.

Vor dem Kopieren **lizenzpflichtiger Software** ist im Rechenzentrum ein Antrag zu stellen, der in der Regel von einem Hochschullehrer des bestellenden Instituts abzuzeichnen ist. Diese Bestätigung entfällt bei der Weitergabe des Programms PCTEX für Studenten im Rahmen der Campuslizenz für dieses Produkt. In jedem Fall ist eine Endbenutzervereinbarung zu unterschreiben, bevor Sie ein lizenzpflichtiges Produkt mit Hilfe von VAMP kopieren können. Die Benutzerkennung für VAMP ist bei Benutzern, die bereits eine Benutzerkennung für den Großrechner besitzen, mit dieser identisch. Andere Benutzer erhalten eine besondere Kennung zugewiesen. Das von VAMP angeforderte Paßwort hat zunächst mit einem eventuell vorhandenen VM-Paßwort nichts zu tun, da VAMP eigene Paßwörter verwaltet.

Zur Kategorie der Lizenz-Programme zählt auch das „Tübinger System von Textverarbeitungs-Programmen“ (TUSTEP), von dem jetzt die neue Version 2.91 vorliegt. Die Erweiterungen betreffen das Programm EINFUEGE und bei den nichtlateinischen Schriften die Hinzunahme der arabischen Schrift für LQ. Installationshinweise und Ergänzungen zum Handbuch stellt Herr Kaspar (Tel. 2473) zum Kopieren zur Verfügung.

Das Textretrieval-System CDS/ISIS liegt jetzt in der Version 2.32 vor. Unterschiede sind in der mitgelieferten Datei READ.ME beschrieben. Sie betreffen neben der Beseitigung kleinerer Fehler die Unterstützung des Mehrbenutzerbetriebs und Erweiterungen der zugehörigen Programmiersprache CDS/ISIS Pascal.

Für Mitte Mai ist die Bereitstellung von TEX30 über VAMP geplant. Das TEX30-Paket wird das bisherige PCTEX-Paket ablösen und ist eine Zusammenstellung aus emTEX und anderen Produkten.

Die Schreibdichte der zu verwendenden $5\frac{1}{4}''$-Disketten und deren Anzahl ist für die einzelnen Programme in den hier gezeigten Tabellen angegeben. Bei PCTEX und SPSS/PC+ handelt es sich um die Minimalzahl für ein funktionsfähiges System. Mit Ausnahme der ersten von VAMP

angeforderten Diskette brauchen die Disketten vorher nicht formatiert zu werden.

PD-Software:

Programm		Disketten	
Name	Version	Dichte	Anzahl
DAD	1.01	2S/2D	1
GVT	2.0	2S/2D	2
MS-DOS-KERMIT	3.0	2S/HD	1
RUMGraph	1.00	2S/2D	1
emTEX	10/90	2S/HD	6
TEXCAD	2.8	2S/2D	1

Lizenz-Software:

Programm		Disketten	
Name	Version	Dichte	Anzahl
CDS/ISIS	2.32	2S/2D	3
KEDIT	4.0	2S/2D	1
NAG FORTRAN	PC 5.0	2S/2D	3
PCTEX	2/93	2S/2D	32
SPSS/PC+	3.0	2S/HD	11
SPSS-Update	3.1	2S/2D	18
StatGraphics	4	2S/2D	10
TUSTEP	2.91	2S/HD	3

Zu den anderen in den Tabellen genannten Produkten sei auf Artikel im infoRUM hingewiesen: DAD wurde in der Ausgabe Nr.3/1990 beschrieben, GVT (Nr. 1/1991), KERMIT und RUMGraph (diese Ausgabe), emTEX(Nr. 4/1990), CDS/ISIS (Nr. 4/1989), KEDIT (Nr. 3/1986), SPSS/PC+ (Nr. 3/1990), PCTEX(Nr. 2/1990).

Programmquerschnitt

von

A. Ahrens

In den Monaten Januar und Februar 1991 ergab sich im Batch-Betrieb auf Jobstep-Basis folgende Verteilung der Programmbenutzung:

Programm	CPU-Zeit in %	Anzahl in %
Backup	0.06	0.28
BMDP	0.03	2.50
MPSX	0.01	0.11
FORTRAN66 Compile	0.08	3.05
FORTRAN66 Execute	8.20	6.44
FORTRAN77 Compile	0.83	11.80
FORTRAN77 Execute	29.00	12.49
Gaussian 82	46.57	1.02
Linkedit	0.11	2.33
Reduce	0.06	0.20
Pascal Compile	0.02	0.02
Pascal Execute	4.99	1.60
PL/I Compile	0.19	2.95
PL/I Execute	2.16	10.85
C Compile	0.00	0.03
C Execute	0.01	0.03
SAS	0.28	1.03
SPSS^X	2.13	9.31
System	3.94	1.58
Utilities	1.34	32.17

Software-Information

Als Software-Information Nr. 21 ist jetzt die Schrift „RUM/GKS“ erschienen, die die am Rechenzentrum implementierte GKS-Version für die Programmiersprachen FORTRAN und PL/I beschreibt. Sie basiert auf den CMS-Hilfedateien, bietet aber darüber hinaus eine allgemeine Einführung mit einer Reihe der wichtigsten Musteranwendungen und zahlreichen grafischen Abbildungen. Das Heft ist zum Preis von 7 DM im Sekretariat des Rechenzentrums erhältlich.

RUM-Tutorial

Fortran 90 (Teil 2)

von

St. Ost

Im zweiten Teil dieser Artikelreihe wird es konkreter. Zu den wichtigen Neuerungen von Fortran 90 gibt es Erklärungen und Beispiele.

Eine Bemerkung vorab. Der Artikel richtet sich vor allem an Fortran-Programmierer. Ich verwende Fortran-spezifische Fachtermini ohne weitere Erläuterungen. Für eine ausführliche Sprachbeschreibung sei auf das Buch von Metcalf/Reid verwiesen[1]. Einige der neuen Sprachelemente sind bereits heute implementiert. Untersuchen Sie doch einmal daraufhin den Fortran-Compiler ihrer Wahl.

Source-Form

Der Zeichensatz wurde um den ganzen ASCII-Zeichensatz erweitert, einschließlich der Kleinbuchstaben. Namen können bis zu 31 Zeichen lang sein, Groß- und Kleinbuchstaben werden nicht unterschieden. Leerzeichen *sind* signifikant, können also nicht mehr an beliebiger Stelle des Quellprogramms stehen, insbesondere nicht in Schlüsselwörtern eingestreut.

Eine Anweisungszeile kann maximal 132 Zeichen lang sein, irgendeine Bindung an bestimmte Spaltenpositionen existiert nicht mehr. Eine Anweisung wird durch ein &-Zeichen am Ende fortgesetzt. Kommentare können zusammen mit einer Anweisung in einer Zeile stehen; was innerhalb einer Zeile rechts vom Ausrufungszeichen ! steht, ist ein Kommentar. Eine Zeile kann mehr als eine Anweisung enthalten; als Trennzeichen dient das Semikolon ;.

Relationale Operatoren

Zu den bisher gebräuchlichen Operatoren werden folgende Alternativen eingeführt.

```
<    .LT.        >    .GT.
<=   .LE.        >=   .GE.
==   .EQ.        /=   .NE.
```

Typdeklaration

Bislang wurden die Eigenschaften eines Objekts an mehr als einer Stelle des Programms erklärt (`COMMON`, `PARAMETER`, `DIMENSION`, `REAL`, ...). Eine neue Form der Typvereinbarung erlaubt es, alle Eigenschaften eines Objekts an einer Stelle zu vereinbaren. Die Anweisung

```
REAL, DIMENSION(5,5), PARAMETER ::  &
      a = (/ (0.0, i = 1, 25) /),   &
      b = (/ (  i, i = 1, 25) /)
```

vereinbart zwei Objekte `a` und `b` als konstante Matrizen vom Typ `REAL`.

Es ist nach wie vor nicht notwendig, jede Variable zu deklarieren. Wer sich trotzdem dazu zwingen will, hat mit

```
IMPLICIT NONE
```

dazu die Möglichkeit.

[1] Metcalf/Reid *Fortran 90 Explained*, Oxford University Press 1990

Einstellbare Datentypen

Angenommen, ein bestimmter Rechner besitzt zwei Arten von ganzen Zahlen, eine normalerweise benutzte (4 Byte lang) und eine kürzere (2 Byte). Die eine Art bekommt den Artenwert (kind type value) 4, die andere den Artenwert 2 zugeordnet. Artenwerte können bei der Typdeklaration verwendet werden, z. B.

```
INTEGER(KIND=4) :: i   ! Voreinstellung
INTEGER(2) :: j        ! halb so gross
```

Für Konstanten gilt das auch. Die Zahl 69 kann auf drei Arten geschrieben werden; 69_4 und 69 sind identisch, 69_2 ist dagegen eine ganzzahlige Konstante, die halb so viel Speicherplatz in Anspruch nimmt. Natürlich ist die explizite Angabe eines Artenwertes nicht unbedingt portabel, besser ist die nachstehende Parametrisierung:

```
INTEGER, PARAMETER :: short=2
INTEGER(short)     :: j
   ⋮
j = 69_short
```

Entsprechende Vorkehrungen können auch für LOGICAL-Datentypen implementiert sein, die den Variablenwert in Bits oder Bytes packen. Für CHARACTER-Daten ebenfalls; hier insbesondere für Zeichensätze, die sich nicht in einem Byte darstellen lassen (z. B. Kanji).

Gleitkommazahlen müssen in wenigstens zwei Arten implementiert sein, entsprechend der heutigen einfachen und doppelten Rechengenauigkeit. Nachstehendes Beispiel definiert einen Vektor, dessen Elemente im Exponentenbereich $10^{\pm 99}$ liegen und mindestens neun dezimale Stellen Rechengenauigkeit aufweisen sollen.

```
INTEGER, PARAMETER ::     &
    long = SELECTED_REAL_KIND(9, 99)
REAL(long), DIMENSION :: a(1024)
a(1) = 1.23456789_long
```

SELECTED_REAL_KIND ist eine spracheigene Funktion und gibt den Artenwert zurück, der den gestellten Anforderungen am nächsten kommt.

Das Pointer-Attribut

Pointer (Zeiger auf Daten) sind in Form eines Attributs in die Sprache eingeführt worden. Die Deklaration eines Objektes mit dem Attribut Pointer vereinbart einen Zeiger auf das Objekt, nicht das Objekt selbst. Es wird kein Speicher reserviert; dieser muß gesondert angefordert werden.

Ein Pointer kann sich auf ein dynamisches Objekt, etwa eine Matrix mit variablen Dimensionsgrenzen, oder auf ein statisches Objekt beziehen. Der Name eines Pointers steht meistens für das Objekt, auf das er zeigt (automatische Dereferenzierung), insbesondere in Ausdrücken und E/A-Anweisungen. Der Wert eines Pointers kann sich durch Zuweisung ändern, allerdings kann er nur auf Datenobjekte zeigen, für die er ursprünglich definiert wurde (strong typed).

```
REAL, DIMENSION(:,:), POINTER :: Pfeil
   ⋮
ALLOCATE(Pfeil(10,10))
   ⋮
Pfeil(1,1) = Pfeil(5,5) + Pfeil(10,10)
```

Pfeil ist ein Pointer auf eine zweidimensionale Matrix vom Typ REAL. Mit der ALLOCATE-Anweisung wird die zugehörige Matrix mit den Dimensionsgrenzen (10,10) angelegt.

Pointer können Teil einer Struktur und vom selben Typ wie die Struktur selbst sein. Es ist also eine rekursive Definition einer Struktur erlaubt, mit deren Hilfe man lineare Listen, Bäume und Graphen beschreiben kann.

```
TYPE Linear_List
  REAL x, y
  TYPE (Linear_List), POINTER :: next
END TYPE Linear_List
```

Pointer werden mit einem Objekt entweder durch eine Pointer-Zuweisung oder über eine ALLOCATE-Anweisung verbunden. Die NULLIFY-Anweisung unterbricht diese Verbindung, der angelegte Speicherplatz dagegen wird über die

DEALLOCATE-Anweisung freigegeben. Es ist daher möglich, daß ein Pointer auf keinen gültigen Speicherplatz verweist, worauf der Programmierer selbst achten muß. Die spracheigene Funktion ASSOCIATED gibt Auskunft, ob eine Beziehung zwischen Pointer und Datenobjekt besteht oder nicht.

CASE

CASE erlaubt die Selektion eines Programmteils in Abhängigkeit von den Werten eines INTEGER-, LOGICAL- oder CHARACTER-Ausdrucks. Ein Beispiel:

```
SELECT CASE(3*i-j)
CASE(0)          ! fuer 0
   :
CASE(2,4:8)      ! fuer 2, 4 bis 8
   :
CASE DEFAULT     ! fuer alle anderen
   :             ! Werte
END SELECT
```

Das CASE-Sprachelement ist funktional dem bisherigen IF ... THEN ... ELSEIF äquivalent, nur deutlich übersichtlicher.

DO-Schleifen

Die DO-Anweisung hat eine neue Form bekommen, die sich mit der bisherigen kombinieren läßt. Die vereinfachte Form ist

```
[name:] DO [control]
        :
        Folge von Anweisungen
        :
        END DO
```

Die in eckigen Klammern stehenden Ausdrücke sind optional. Fehlt die Angabe control, so wird eine Endlos-Schleife angenommen. Fehlt sie nicht, so entspricht sie der bisherigen Schleifenbedingung, allerdings müssen Schleifenvariable, Anfangs- und Endwert sowie das Inkrement vom Typ INTEGER sein.

Eine DO-Schleife kann einen Namen bekommen, auf den man sich bei der CYCLE- und der EXIT-Anweisung beziehen kann. CYCLE beginnt eine neue Iteration, EXIT beendet die Verarbeitung der DO-Schleife.

Eine DO-WHILE-Schleife ist nachträglich in die Sprache aufgenommen worden; sie ist aber redundant, da die gleiche Wirkung auch mittels DO und EXIT erreicht werden kann.

Ausblick

Der dritte (und letzte) Teil der Fortran-90-Übersicht folgt dann in der nächsten infoRUM-Ausgabe. Dann geht es um Vektoroperationen, Unterprogramme und selbstdefinierte Datentypen.

RUMgraph – Einbindung von Pixel-Bildern in ein TEX-Dokument (Teil 1)

von

H.-W. Kisker

Eine neue Software bringt die für TEX so typische Geräteunabhängigkeit auch bei Bildern.

Der klassische Weg, in TEX-Dokumente Bilder einzubinden, ist die Verwendung des *special*-Kommandos. Dieses wird nicht von TEX selbst, sondern ausschließlich vom DVI-Treiber abgearbeitet. Ich habe in den vergangenen Jahren einige DVI-Treiber für verschiedene Drucker geschrieben, um Grafiken benutzen zu können. Nach meinem Dafürhalten kranken aber alle Lösungen, die sich auf das *special*-Kommando abstützen, in zweierlei Hinsicht.

Wie sein Name schon andeutet, ist die Bedeutung des *special*-Kommando in keiner Weise festgelegt. Jeder Treiber, der ein solches Kommando implementiert, kann sowohl Syntax als auch Semantik selbst festlegen. Hiervon wird von den

Entwicklern der DVI-Treiber ungehemmt Gebrauch gemacht. Auch die momentanen Normungsbestrebungen werden an dieser Situation kaum etwas ändern. Jeder so entwickelte Vorschlag hat nämlich nicht den bindenden Charakter eines TEX-Kommandos, sondern ist als Empfehlung zu verstehen, die man befolgen, aber auch verwerfen kann. Die einzige allgemein anerkannte Autorität in der TEX-Gemeinde, die hier etwas ändern könnte, ist Donald Knuth selbst. Würde er TEX mit Grafikelementen versehen, so wäre die Frage der Implementierung jeder Diskussion entzogen. Ich persönlich würde mir eine solche Erweiterung wünschen.

Der zweite Gesichtspunkt, der die Verwendung des *special*-Kommandos unhandlich macht, ist seine Nichtbeachtung durch TEX. TEX hat keine Vorstellungen über die Abmessungen einer Grafik, die durch einen solchen Befehl in ein Dokument aufgenommen wird. Für den Benutzer bedeutet dies, daß er das Bild in eine Schale (z. B. *picture*-Umgebung) einbetten muß, um TEX indirekt die Möglichkeit zu geben, das grafische Element zu berücksichtigen. Dieses Verfahren schafft neue Fehlerquellen und zusätzlichen Pflegeaufwand. So können z. B. die Abmessungen eines Bildes falsch eingeschätzt werden, und auf eine Änderung der Grafik muß mit einer Änderung der Parameter für die Schale aus TEX-Kommandos reagiert werden.

Beide Hindernisse haben mich schon vor einiger Zeit dazu veranlaßt, Lösungen mit dem *special*-Kommando nicht weiter zu verfolgen. Ich habe für meine Belange stattdessen einen anderen Weg realisiert. Ausgangspunkt war die Feststellung, daß TEX standardgemäß mit „Grafiken" umzugehen weiß, nämlich mit den Zeichen seiner Zeichensätze. Aus diesem Blickwinkel betrachtet, hält TEX für jede dieser „Grafiken" ein Rechteck auf der Seite frei, in das dann der DVI-Treiber das „Bild" des Zeichens hineinmalt. Die Zeichen werden dabei für pixelorientierte Drucker als Bitmap vorgehalten, für PostScript-fähige Drucker z. B. auch als das Aussehen beschreibende Algorithmen.

Ich habe mich bei meinen Entwicklungen auf pixelorientierte Drucker beschränkt. Damit reduziert sich die Aufgabe, Bilder für TEX aufzubereiten, auf die Umwandlung von Grafik-File-Formaten in eines der von TEX unterstützten Zeichensatzformate PXL oder PK. Der Standardweg, solche Zeichensätze zu erzeugen, führt über METAFONT und GFtoPX. Lösungen, die hier ansetzen und z. B. HPGL-Files in *mf*-Files umsetzen, sind bereits als kommerzielle Produkte verfügbar. Allerdings liegen viele Grafiken nicht in Vektor-Form (wie HPGL) vor, sondern als Pixel-Files. Insbesondere über einen Scanner gewonnene Bilder werden fast immer in einem PCX-, TIFF- oder GIF-File abgelegt. Ich habe deshalb dem direkten Weg zur Erzeugung der Zeichensätze den Vorzug gegeben. Das unten vorgestellte Programm pxlgraph wandelt ADI- und PCX-Grafik-Files in TEX-PXL-Files um. Die Implementierung des ADI-Formats ist als Abdeckung meines persönlichen Bedarfs zu sehen: ich benutze bevorzugt AutoCAD zur Erzeugung meiner Bilder. Die größere Bedeutung kommt aber sicherlich dem weit verbreiteten PCX-Format der Firma ZSoft zu. Auf die Implementierung anderer Grafikformate habe ich verzichtet, da es sowohl im kommerziellen als auch im Public-Domain-Bereich Programme gibt, die nahezu jedes Grafikformat – sowohl pixel- als auch vektororientiert – nach PCX umwandeln. Ich habe bereits erfolgreich GIF-, MSP-, TIFF-, HPGL-, PCL-, EPS-, WPG-, IGES- und DXF-Formate verwendet.

Aufgabe des Programms

Das Programm pxlgraph wandelt Pixel-Bilder im ADI- oder PCX-Format in TEX-PXL-Zeichensätze um. Für jeden Buchstaben des neu zu schaffenden Zeichensatzes wird dabei ein eigenes Grafik-File verwendet.

Eingabe für das Programm sind dementsprechend neben einer Reihe von Angaben, die den Zeichensatz beschreiben, ein oder mehrere Grafik-Files für die Buchstaben. Als Ergebnis wird ein PXL-File mit zugehörigem TFM-File erzeugt. Außerdem wird ein sogenanntes *Log-File* geschrieben, das ausführliche Information über den jeweiligen Programmlauf enthält. Auf Anforderung kann ein TEX-Input-File bereitgestellt

werden, das den Zugriff auf den Zeichensatz und seine Zeichen erleichtert.

Das Programm ist ablauffähig auf IBM-kompatiblen PCs. Es läuft unter den Betriebssystemen MS-DOS und OS/2.

Der Aufruf von RUMgraph

Das Programm RUMgraph wird in der üblichen Weise durch eine Eingabe der folgenden Form aufgerufen:

`rumgraf` *option ...*

Die Optionen beginnen dabei mit den Zeichen / oder –. Unmittelbar danach folgt ein einzelner Buchstabe: die Optionskennung. Welche Buchstaben zulässig sind und welche Bedeutung sie haben, wird weiter unten erläutert. Groß- und Kleinbuchstaben werden *nicht* unterschieden. An die Optionskennung schließt sich der Wert der Option an. Zwischen Optionskennung und Wert kann als Trenner eines der Zeichen : oder = eingeschoben werden. Leerzeichen sind als Trenner nicht zulässig. Enthält der Wert der Option ein Leerzeichen, so ist er in Hochkommata ' ' einzuschließen. Alle Angaben, die nicht mit / oder – beginnen, werden als Kommentar betrachtet und dementsprechend ignoriert.

Die Optionen gliedern sich in drei Gruppen:

- Die erste Gruppe spezifiziert die Parameter, die für den gesamten Zeichensatz gelten (z. B. Auflösung). Sie können sinnvollerweise nur einmal angegeben werden. Erfolgt eine mehrfache Angabe, so hat nur der letzte Wert Gültigkeit.
- Die zweite Gruppe beschreibt die einzelnen Zeichen (z. B. metrische Angaben). Eine einem Zeichen zugeordnete Gruppe von Optionen wird eingeleitet durch die c-Option (character number), die ein bestimmtes Zeichen festlegt. Alle Angaben bis zur nächsten c-Option werden diesem Zeichen zugeordnet. Bei Mehrfachangaben behält wieder nur die letzte Angabe einer Gruppe ihre Gültigkeit.
- Die dritte Gruppe enthält nur eine einzelne Option, und zwar die m-Option. Über sie wird der Name eines sogenannten *makefiles* angegeben. In ihm können beliebige Optionen abgelegt sein. Die Angabe einer m-Option bewirkt, daß die in dem *makefile* enthaltenen Optionsangaben in die Reihe der Optionen der Kommandozeile eingeschoben und abgearbeitet werden. Alle Optionen können in einem *makefile* verwendet werden, insbesondere auch Kommentare.

Optionen der Gruppe 1: Angaben zum Zeichensatz

/P:*ident*

Mit der p-Option wird eine Kennung für das zu erzeugende Bild festgelegt. Sie wird benutzt für den Namen des

- PXL-Files (ident.PXL),
- TFM-Files (ident.TFM),
- TEX-Input-Files (ident.TEX) und des
- Log-Files (ident.RGL).

Die Kennung muß aus 1 bis 8 Buchstaben bestehen.

Beispiel: /P:SINGA

/R:*Resolution*

Die r-Option gibt die verwendete Auflösung der Bitmap an.

Maßeinheit: dpi (= dots per inch)

Beispiel: /R:300

Optionen der Gruppe 2: Angaben zu den einzelnen Zeichen

/C:*Character*

Mit der c-Option wird der Beginn eines Parametersatzes für das angegebene Zeichen festgelegt. Alle Optionen der Gruppe 2 werden bis zur nächsten c-Option auf dieses Zeichen bezogen. Als Wert kann entweder ein Buchstabe oder eine Zahl von 0 bis 127 angegeben werden.

Beispiele: /C:A oder /C:65

/N:*Name*

Jedem Buchstaben kann über die n-Option ein Name zugeordnet werden. Der Name dient zur Generierung eines TEX-Kommandos im TEX-Input-File. Wird das Input-File benutzt, so kann der Buchstabe – und somit auch das Bild – durch Angabe dieses Kommandos in das Dokument eingefügt werden. Fehlt die n-Option, so wird für diesen Buchstaben kein Kommando erzeugt. Wird für keinen der Buchstaben ein Name angegeben, so wird die Erzeugung des TEX-Input-Files gänzlich unterdrückt. Ein Name kann aus einer beliebigen Anzahl von Buchstaben zusammengesetzt sein.

Beispiel: /N:Aleph

/T:*Type*

Die t-Option bezeichnet das Format des zu dem Buchstaben gehörenden Grafik-Files. Als Werte können die Kennungen ADI und PCX angegeben werden. Für jeden Buchstaben kann ein anderer File-Typ angegeben werden.

Beispiel: /T:PCX

/F:*Filename*

Über die f-Option wird der Name des Grafik-Files angegeben, das für die Erzeugung der Bitmap des Zeichens verwendet werden soll. Es kann ein beliebiger Pfadname angegeben werden.

Beispiel: /F:aleph.pcx

/D:*Depth* /H:*Height* /W:*Width*

Die metrischen Angaben für das TFM-File werden über die d-Option (*depth*), h-Option (*height*) und w-Option (*width*) spezifiziert. Die Bedeutung dieser und der anderen metrischen Werte wird weiter unten erläutert. Fehlen die Optionen, so wird *width* gleich der Breite *x size* der Bitmap, *height* gleich der Höhe *y size* der Bitmap und *depth* gleich Null gesetzt. Alle Angaben sind in Pica-Punkten *pt* zu machen.

Maßeinheit: pt (1 inch = 72.72 pt)

Beispiel: /W:20 /H:24 /D:8

/X:*Xoffset* /Y:*Yoffset*

Über die x-Option (*x offset*) und die y-Option (*y offset*) können die metrischen Angaben für das PXL-File festgelegt werden. Die Größen *x size* und *y size* entnimmt pxmgraph direkt dem über die f-Option bestimmten Grafik-File. Fehlen die Angaben, so wird *x offset* gleich 0 und *y offset* gleich *y size* gesetzt.

Maßeinheit: pixel

Beispiel: /X:–5 /Y:22

/S:*Splitcount*

Werden Bilder in Buchstaben umgewandelt, so entstehen häufig äußerst große Buchstaben. Eine Zeichnung, die eine ganze DIN-A4-Seite einnimmt, ist keine Besonderheit. Viele Treiber (und auch viele PXtoPK-Implementierungen) können so große Zeichen nicht handhaben. Deshalb besteht über die s-Option die Möglichkeit, das Bild in mehrere Teilbilder in Form von vertikalen Streifen zu zerschneiden. Diese Streifen werden dann einzeln in Buchstaben umgewandelt und unbelegten Codes des Zeichensatzes zugewiesen. Die metrischen Angaben im TFM- und PXL-File werden dabei so angepaßt, daß alle Buchstaben, wenn sie unmittelbar aufeinander folgen, wieder das Gesamtbild ergeben.

Maßeinheit: Bytes

Beispiel: /S:20

/I
Mit Hilfe der i-Option kann ein Bild invertiert werden.

Beispiel: /I

Die Standardwerte der metrischen Angaben sind so gewählt, daß sowohl aus der Sicht von TEX als auch aus der des DVI-Treibers der Referenzpunkt des Zeichens stets in der linken unteren Ecke liegt und daß um die Bitmap herum keinerlei Rand bleibt. Wird pbmgraph zur Einbindung von Bildern in ein Dokument genutzt, so ist diese Setzung naheliegend.

Optionen der Gruppe 3: Umlenken der Eingabe auf ein *makefile*

/M:*Makefile*

Über die m-Option kann pbmgraph angewiesen werden, seine Optionen aus einer Datei zu lesen. Insbesondere wenn ein zu erzeugender Zeichensatz mehrere Buchstaben umfaßt, ist die Einführung eines *makefiles* unumgänglich. Die Anzahl der Optionen, die in der Kommandozeile möglich sind, ist eben deutlich beschränkt. Ein *makefile* kann alle Optionen enthalten. Wird allerdings in einem *makefile* eine m-option angegeben, so werden alle folgenden Optionen ignoriert. Ein *makefile* kann deshalb innerhalb eines *makefiles* sinnvoll nur mit der letzten Option angegeben werden.

Beispiel: /M:Singa

Die Verwaltung von Zeichensätzen durch TEX

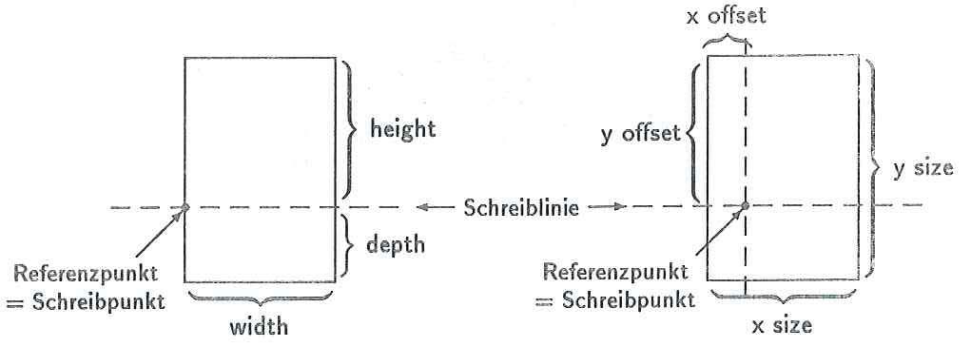
Ein Zeichensatz, wie er von TEX verwendet wird, enthält grundsätzlich zwei Arten von Angaben:

- metrische Angaben wie Breite und Höhe der Zeichen und
- die grafische Beschreibung des Zeichens selbst – meistens wie bei PXL- und PK-Files in Form einer Bitmap.

Die Angaben für jeden Zeichensatz sind auf zwei Dateien verteilt: das TFM-File und das PXL-File. Die Angaben in beiden Files sind teilweise redundant. Etwas vereinfacht kann man aber sagen, daß das TFM-File metrische Angaben enthält, die den Platz beschreiben, der für das Zeichen auf der Seite reserviert werden soll, und daß das PXL-File die Bitmap beinhaltet, die zur eigentlichen Darstellung des Zeichens benutzt wird. Entsprechend dieser Aufteilung werden bei der Erstellung eines Dokuments mit TEX die Angaben eines Zeichensatzes von unterschiedlichen Programmen ausgewertet.

Das eigentliche TEX-Programm benutzt nur die metrischen Angaben aus dem TFM-File. Aus der Sicht von TEX sind Buchstaben einfach Rechtecke (siehe Abb. 1). Zur Positionierung dieser Rechtecke wird eine fiktive *Schreiblinie* und auf dieser ein fiktiver *Schreibpunkt* verwaltet. Die Buchstabenrechtecke werden auf dieser Schreiblinie so positioniert, daß zum einen ihr linker Rand durch den Schreibpunkt hindurchgeht und daß sie zum anderen eine vorgegebene Höhe (*height*) über der Linie und eine vorgegebene Tiefe (*depth*) unter der Linie hinausragen. Die Schreibstelle wird von TEX nach der Positionierung des Zeichens um die Breite *width* des Zeichens verschoben. Man kann auch sagen, daß die Angaben von *height* und *depth* auf dem linken Rand des Zeichens einen *Referenzpunkt* festlegen, der von TEX mit dem Schreibpunkt zur Deckung gebracht wird. TEX berücksichtigt in keiner Weise das durch die Bitmap festgelegte grafische Aussehen eines Zeichens. Es hält nur den Platz für diese Bitmap frei. Das *Ausmalen* dieses Platzes ist Aufgabe des DVI-Treibers.

Ein DVI-Treiber überträgt also die im PXL-File (oder auch PK-File) enthaltene Bitmap auf die Seite. Dabei werden auch von ihm metrische Angaben, die sich aber auf die Abmessungen der Bitmap beziehen, ausgewertet. Konkret sind dies vier Werte. Zum einen existieren auch hier Angaben zur Höhe (*y size*) und Breite (*x size*) eines jeden Zeichens. Sie spiegeln das engste Rechteck wieder, das um die Bitmap gezogen werden kann. Zum anderen wird mit einem horizontalen (*x offset*) und vertikalen (*y offset*) Versatz ebenfalls



Die Sicht von TEX — Die Sicht des Treibers

Abbildung 1: Buchstabenrechteck

ein Referenzpunkt definiert. Da beide Offset-Angaben auch negativ sein können, kann der Referenzpunkt beliebig außerhalb der Bitmap liegen. Ein DVI-Treiber verwaltet – wie das TEX-Progamm selbst – Schreiblinie und Schreibpunkt und positioniert Zeichen, indem Schreibpunkt und Referenzpunkt der Bitmap zur Deckung gebracht werden. Bei Verschiebung der Schreibstelle wird allerdings der gleiche Wert für die Breite des Zeichens verwendet wie vom TEX-Programm.

Prinzipiell sind die Angaben im TFM- und PXL-File unabhängig voneinander. In der Praxis werden die Angaben aber natürlich aufeinander abgestimmt verwendet. Der von TEX reservierte Platz ist meistens so gewählt, daß um den gedruckten Buchstaben herum ein ausreichender Abstand zu benachbarten Zeichen entsteht. Erforderlich ist dies allerdings nicht; so kann die vom DVI-Treiber auf die Seite übertragene Bitmap durchaus außerhalb des von TEX freigehaltenen Platzes zu liegen kommen.

Arbeitsweise von PUMgraph

Bei der Erzeugung von Zeichen mit dem Programm PUMgraph können die Angaben *width*, *height* und *depth* für das TFM-File und *x offset* und *y offset* für das PXL-File vom Benutzer festgelegt werden. Die fehlenden Größen *x size* und *y size* für das PXL-File entnimmt PUMgraph direkt dem Ursprungs-Grafik-File.

Die oben beschriebene Möglichkeit, Bitmaps auch außerhalb des von TEX freigehaltenen Platzes zu positionieren, wird genutzt, wenn ein großes Bild in mehrere Buchstaben zerlegt werden muß. Dabei wird nämlich der Referenzpunkt für jeden Teilbuchstaben an die Stelle gelegt, die durch den Referenzpunkt des Gesamtbildes bestimmt ist. Die Breiten der ersten Teilbilder werden sämtlich auf Null gesetzt und nur die Breite des letzten Teilbuchstabens wird gleich der Weite des Gesamtbildes gewählt. Zur Erläuterung werde Abb. 2 betrachtet.

Der Referenzpunkt des Bildes ist die linke untere Ecke. Wird es mit einer Setzung /S:3 in ein PXL-File umgewandelt, so entstehen drei Teilbuchstaben mit Referenzpunkten wie in Abb. 3.

Werden die drei Buchstaben unmittelbar nacheinander ausgegeben, so entsteht wieder das Gesamtbild aus Abb. 2.

Natürlich hätte auch die Möglichkeit bestanden, für die Teilbuchstaben den Referenzpunkt stets auf die jeweilige linke untere Ecke zu legen und die Breite gleich der jeweiligen Pixel-Ausdehnung zu wählen. Diese Lösung birgt aber eine Gefahr in sich. Für die Positionierung seiner Buchstaben auf der Schreiblinie verwaltet ein

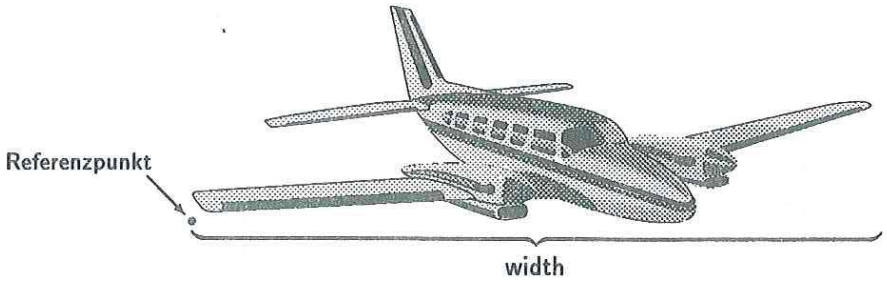


Abbildung 2: Gesamtbild

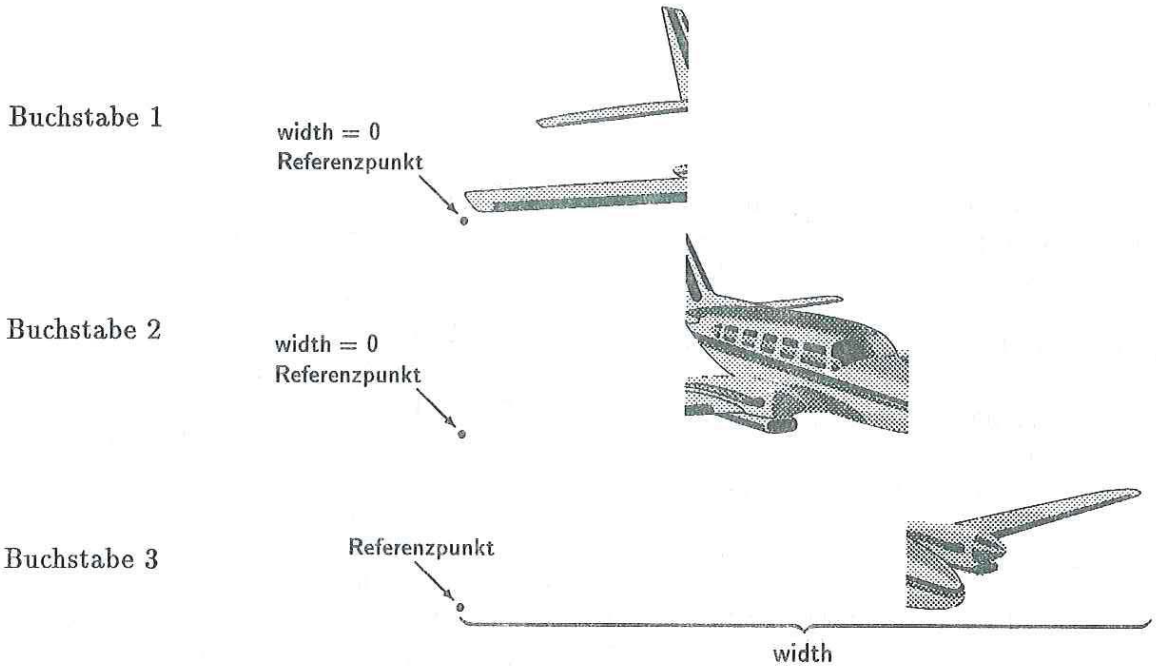


Abbildung 3: Teilbuchstaben

DVI-Treiber zwei Koordinatenwerte: zum einen die idealen hochgenauen TEX-Koordinaten und zum anderen die realen durch die Auflösung bestimmten Pixel-Koordinaten. Durch Rundungsfehler driften diese beiden Werte auseinander. Deswegen überprüft der Treiber bei jeder Positionierung, ob die Differenz eine vorgegebene Schwelle (Stichwort: *max drift*) überschreitet. Ist dies der Fall, so wird die Position um ein oder mehrere Pixel korrigiert. Fällt nun diese Korrektur zwischen zwei zu einem Bild gehörende Teilbuchstaben, so entstehen entweder weiße Streifen im Bild oder zwei Teilbilder überlappen sich. Beides ist nicht tolerierbar. Die Lösung mit der Breite Null für die ersten Buchstaben vermeidet dieses Problem.

Anmerkungen

Das Programm **rumgraph** entfernt sämtlichen Leerraum rund um das Bild. Übrig bleibt ein Rechteck, das auf jeder Seite mit wenigstens einem schwarzen Punkt besetzt ist. Soll um das Bild herum Leerraum vorgesehen werden, um Platz zu den Nachbarzeichen zu gewinnen, so ist

dies über die Optionen für die metrischen Werte (x, y, w, h und d) zu erreichen.

Wird ein großes Bild in Teilbilder zerlegt, so werden diese fortlaufend freien Zeichen zugeordnet. Die Zuordnung beginnt mit der in der c-Option angegebenen Zeichennummer. Wird ein Zeichen erreicht, das bereits belegt ist, so wird es übersprungen. Wird das Zeichen mit der Nummer 127 erreicht, so wird anschließend mit dem Zeichen mit der Nummer 0 fortgefahren. Die von RUMgraph vorgenommene Zuordnung kann sowohl dem Log-File (Erweiterung: RGL) als auch dem TEX-Input-File entnommen werden.

Ausblick auf Teil 2

Der zweite Teil dieses Artikels beschäftigt sich mit den vielfältigen Quellen für Bilder und deren Aufbereitung für TEX und andere Programme. Als kleiner Vorgeschmack seien einige typische Bilder angefügt.

Mein Lieblingslöwe:

Ein Photo:

Einige Hieroglyphen:

Aleph dj f l m w ah gm ha3 hn

RUM-Grafik

Ausgabe von PCX-Grafiken auf dem Thermo-Transfer-Drucker

von

A. Jankrift

Gut gelungene Farbgrafiken können jetzt nicht nur auf dem PC-Bildschirm bewundert werden, sondern auch auf einem Blatt Papier.

Oftmals besteht der Wunsch, daß Farbgrafiken, die auf dem PC erzeugt worden sind, auf einem hochwertigen Farbplotter ausgegeben werden sollen, z. B. auf dem Thermo-Transfer-Drucker des Rechenzentrums. Dazu müssen die Bilddaten zunächst zum Großrechner übertragen werden. Dies bringt aber i. a. Probleme mit sich. Die verschiedenen Grafik-Formate, die sich mittlerweile im PC-Bereich etabliert haben, können häufig nicht im Großrechnerbereich „verstanden" werden. Aus diesem Grunde mußten wir uns auf ein bestimmtes Grafik-Format festlegen. Die Entscheidung fiel auf das PCX-Format, das von vielen Produkten unterstützt wird. Grafiken dieses Formats können also nun auf dem Thermo-Transfer-Drucker ausgegeben werden, und zwar mit Hilfe des COLPLOT-Kommandos. Im folgenden werden die dazu nötigen Schritte beschrieben.

Erzeugen einer PCX-Grafik

PCX-Grafiken kann man z. B. mit dem Programm Paintbrush erstellen. Eine andere Möglichkeit besteht darin, Bildvorlagen, z. B. Fotos, mit Hilfe eines Scanners einzulesen und in eine Datei vom Typ PCX zu speichern. Im Rechenzentrum steht dazu ein Grafik-Scanner zur Verfügung. Das dafür benötigte Programm heißt Le Color und läuft unter Microsoft Windows. Zu beachten ist dabei, daß die Bildvorlagen nicht zu groß gewählt werden dürfen, da der Speicherplatzbedarf für farbige Bilder sehr hoch ist. Für eine DIN-A4-Vorlage können bis zu 20 MB benötigt werden. Besitzt man nun eine Grafik in einem anderen Format, z. B. eine TIFF- oder WPG-Datei, so kann sie mit Hilfe geeigneter Programme in das PCX-Format umgewandelt werden.

PCX-Datei-Transfer zum Großrechner

Dateien von der DOS-Ebene können z. B. mit Hilfe des Programms FTP zum Großrechner übertragen werden. Mit dem Kommando

```
ftp host
```

wird der Zugang zum CMS ermöglicht. `host` ersetzt man durch den Namen der gewünschten Maschine (a oder c). Anschließend wird nach der Benutzerkennung und nach dem Paßwort gefragt. Ist die Verbindung zum CMS erfolgreich zustandegekommen, so muß zunächst festgelegt werden, in welchem Satzformat die Datei auf der Minidisk abgespeichert werden soll. Dazu gibt man den Befehl

```
server site i recfm(f) lrecl(80)
```

ein. Die tatsächliche Übertragung geschieht durch das Kommando

```
iput fn1.ft1 [fn2.ft2.fm2]
```

`fn1.ft1` ersetzt man durch den Namen der Quelldatei und `fn2.ft2.fm2` durch den Namen der Zieldatei. Dabei ist zu beachten, daß im CMS-Dateinamen anstelle von Leerzeichen Punkte anzugeben sind. Gibt man keinen Namen für die Zieldatei an, so wird die Bezeichnung der Quelldatei übernommen. Hat man z. B. auf dem PC eine Datei BILD.PCX erzeugt, so wird sie durch

```
iput bild.pcx
```

zum Großrechner übertragen und dort unter der Bezeichnung BILD PCX A gespeichert. Nachdem die Datei übertragen worden ist, verläßt man das Programm FTP mit

```
quit
```

Umwandlung der PCX-Datei in eine COLPLOT-Datei

Nachdem die Grafik-Datei zum Großrechner übertragen worden ist, kann sie dort mit dem CMS-Kommando

```
PCXTOCOL fname (Option
```

in eine COLPLOT-Datei umgewandelt werden. Der Filetyp der zu bearbeitenden Datei muß PCX lauten. Nach erfolgreicher Ausführung des Befehls hat man zwei neue Dateien erhalten. Sie haben den gleichen Namen wie die ursprüngliche Datei. Nur der Dateityp lautet anders. Hat man z. B. den Befehl `PCXTOCOL BILD` eingegeben, so hat man anschließend die neuen Dateien BILD MAT und BILD COL. Die eine Datei enthält die Bilddaten und die andere die Definition der Farben. Sie werden dazu benötigt, um die Grafik mit Hilfe des COLPLOT-Kommandos auf dem Farbplotter auszugeben. Es fehlt noch die Erklärung der Options-Angabe. Gibt man hier den Buchstaben R an, so wird die ursprüngliche Grafik um 90 Grad im Uhrzeigersinn gedreht.

Evtl. kann es vorkommen, daß man nicht mehr genügend Speicherplatz für die Dateien auf der Minidisk hat. In diesem Fall bietet sich das DAYDISK-Kommando an (s. infoyum Nr. 4/1990). Damit kann man für einen ganzen Tag zusätzlichen Plattenspeicherplatz erhalten. Überträgt man die ursprüngliche Grafik-Datei auf diese Platte, so muß das PCXTOCOL-Kommando wie folgt aufgerufen werden

```
PCXTOCOL fname PCX fm (Option
```

Für `fm` muß der Dateimodus der neuen Platte angegeben werden (z. B. T). Die beiden neu erzeugten Dateien besitzen dann auch den angegebenen Dateimodus.

Modifikation und Ausgabe der Grafik

Häufig besteht der Wunsch, daß die Grafiken vor dem Ausdruck noch modifiziert werden sollen, z. B. möchte man die Bilder beschriften. Dazu kann man zu den Dateien MAT und COL noch eine Klartextdatei TXT erzeugen. In ihr kann die Art der Beschriftung festgelegt werden (s. infoyum Nr. 2/1987). Außerdem möchte man vielleicht die Grafiken vor dem Ausdruck noch einmal an einem Farbrasterterminal (IBM 5080 bzw. IBM 3192) anschauen. Hierfür steht das CMS-Kommando COLPLOT zur Verfügung. Falls man eine Grafik-Datei mit Namen BILD besitzt, so kann der Aufruf des Befehls wie folgt aussehen:

```
COLPLOT BILD (L=30,C=50,S=4
```

Die Parameter `L`, `C` und `S` beziehen sich auf die Größe des Bildes. `L` steht für linecount und `C` für columncount. Hier spezifiziert man die Zeilen- und Spaltenanzahl des Bildes. Die maximalen Werte dafür wurden bei der Ausführung des Kommandos PCXTOCOL ausgegeben. Mit dem Parameter `S` variiert man die Größe der Ausgabe. Zu beachten ist dabei, daß bestimmte Werte nicht überschritten werden dürfen (s. dazu die Artikel in infoyum Nr. 4/1988 und 2/1990). Eine genauere Beschreibung kann man auch mit dem Befehl `HELP COLPLOT` erhalten. Der anschließende Ausdruck der Grafik geschieht mit Hilfe der JCL-Prozedur COLPLOT. Dazu werden zunächst die Dateien MAT, COL bzw. TXT zum MVS übertragen. Dies geschieht mit Hilfe des Befehls `SENDMVS`, z. B.

```
SENDMVS BILD MAT
```

Der Job zum Ausdruck der Grafik BILD im MVS kann folgende Gestalt haben:

```
//XYZ12COL JOB (PROD,box),name,
// PASSWORD=(<PASSWORD>)
/*JOBPARM T=100,L=42
// EXEC COLPLOT,PARME='L=30,C=50,S=4'
//MAT DD DSN=XYZ12.BILD.MAT,DISP=SHR
//COL DD DSN=XYZ12.BILD.COL,DISP=SHR
```

Der fertige Ausdruck liegt im Dispatch bereit.

RUM-Lehre

Lehrveranstaltungen im Sommersemester 1991

Einführende Lehrveranstaltungen

320090	Programmieren in FORTRAN Mi 15–17 Hörsaal: M4, Beginn: 24.4.1991	*Grote, M.*
320105	Programmieren in Pascal Di 13–15 Hörsaal: M3, Beginn: 16.4.1991	*Mertz, K.-B.*
320110	Textverarbeitung auf Mikrorechnern[1] Di 13–15 Hörsaal: M5, Beginn: 23.4.1991	*Kamp, H.*
320124	Computerunterstütztes Publizieren mit LaTeX Mi 9–11 Hörsaal: M4, Beginn: 24.4.1991	*Benduhn-Mertz, A.*
320139	Statistische Datenanalyse mit dem Programmsystem SAS Mo 15–17 Hörsaal: M4, Beginn: 15.4.1991	*Zörkendörfer, S.*

Weiterführende Lehrveranstaltungen

320143	Datenstrukturen und Programmierverfahren in Pascal Do 15–17 Hörsaal: M4, Beginn: 18.4.1991	*Bosse, W.*
320158	Objektorientiertes Programmieren mit C++ Do 11–13 Hörsaal: M4, Beginn: 18.4.1991	*Mersch, R.*
320162	Grafische Methoden in der Datenanalyse Mi 13–15 Hörsaal: M4, Beginn: 24.4.1991	*Süselbeck, B.*
320177	Unix Di 13–15 Hörsaal: M4, Beginn: 23.4.1991	*Ost, St.*
320181	Mikrorechner-Praktikum Mo 13–15 CIP-Raum Rechenzentrum, Beginn: 22.4.1991	*Kisker, H.-W.*

[1] Wegen der Begrenzung der Teilnehmerzahl ist für diese Lehrveranstaltung eine frühzeitige Anmeldung im Dispatch des Rechenzentrums erforderlich.

320196	Rechnernetze und ihre Anwendungen Do 13–15 Hörsaal: M4, Beginn: 18.4.1991	*Richter, G./* *Speer, M.*
320200	Betriebssysteme Di 15–17 Hörsaal: M4, Beginn: 23.4.1991	*Neukäter, B.*
320215	Kolloquium über Themen der Informatik Fr 13–15 Hörsaal: M4	*Held, W./* *die wiss. Mitarbeiter* *des Rechenzentrums*
320220	Anleitung zum Einsatz der EDV bei wissenschaftlichen Arbeiten nach Vereinbarung	*die wiss. Mitarbeiter* *des Rechenzentrums*