Forschungs- und Prototypsysteme

9. Delisle, N.M., Schwartz, M.D.: Neptune: A hypertext system for software development environment. Database Eng. *10,* No.1, 54–59 (1987)
10. Hara, Y., Kaneko, A.: A new multimedia electronic book and its functional capabilities. Proc. RIAO'88, M.I.T. Cambridge (MA), März 1988, pp.114–123
11. Halasz, F.G.: NoteCards: An experimental environment for authoring and idea processing. Proc. BTW'87, Darmstadt, April 1987, pp.56–67
12. Halasz, F.G., Moran, T.P., Randall, R.N.: NoteCards in a nutshell. Proc. ACM Conf. CHI/GI'87, Toronto 1987, pp.45–52
13. Meyrowitz, N.: Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. Proc. OOPSLA'86, Spec. Issue of SIGPLAN Not. September 1986, pp.186–201
14. Trigg, R.H., Weiser, M.: TEXTNET: A network-based approach to text handling. ACM ToOIS *4,* No.1, 1–23 (1986)
15. Smith, K.E., Zdonik, S.B.: Intermedia: A case study of the differences between relational and object-oriented database systems. Proc. OOPSLA'87, Spec. Issue of SIGPLAN Not. Oktober 1987, pp.452–465
16. Yankelovich, N., Haan, B.J., Meyrowitz, N., Drucker, S.M.: Intermedia: The Concept of a Seamless Information Environment. IEEE Comput. *21,* No.1, 81–96 (1988)

Repräsentations- und Darstellungsprobleme

17. Feiner, S.: Seeing the forest for the trees. Proc. ACM COIS'88, Palo Alto (CA), März 1988, pp.205–212
18. Foss, C.L.: Effective browsing in hypertext systems. Proc. RIAO'88, M.I.T. Cambridge (MA), März 1988, pp.82–98
19. Hofmann, M., Langendörfer, H.: Interaktives Navigieren mit dem Browser des Hypertextsystems CONCORDE. Notizen zu interaktiven Systemen, Nr.17, Februar 1989, pp.83–88
20. Komorowski, H.J., Greenes, R.A., Barr, C., Pattison-Gordon, E.: Browsing and authoring tools for a unified medical language system. Proc. RIAO'88, M.I.T. Cambridge (MA), März 1988, pp.624–641
21. Marchionini, G., Shneiderman, B.: Finding facts vs. browsing in hypertext systems. IEEE Comput. *21,* No.1, 70–80 (1988)
22. Schwarz, M., Delisle, N.: Contexts – a partitioning-concept for hypertext. ACM ToOIS *5,* No.2, 168–186 (1987)

Overflow

Informatik Spektrum

Informatik-Spektrum (1989) 12: 220–221

In dieser Rubrik werden „unkonventionelle" Beiträge veröffentlicht: kurze, prägnante Texte, die aus irgendeinem Grund keine konventionellen Artikel sind, aber dennoch wissenschaftlich-technisches Interesse verdienen.

Die Attribute „konventionell" und „unkonventionell" sind dabei durchaus wertneutral gemeint. Wir verhehlen aber nicht, daß wir von unkonventionellen Texten ein Quentchen mehr Esprit erwarten, als ihn die Texte des wissenschaftlichen Alltags haben (können). In unserem Idealtext wäre ein eleganter Gedankengang ebenso geistreich wie humorig präsentiert.

Die Leser des *Informatik-Sepktrums* sind eingeladen, Beiträge für diese Rubrik einzureichen; auch besonders gelungene Lösungen zu Aufgaben, die hier gestellt werden, sind willkommen. Bitte senden Sie Ihre Texte an den für diese Rubrik verantwortlichen Herausgeber, Prof. Dr. J. Nievergelt, Institut für Informatik, ETH, CH-8092 Zürich, oder an die Redaktion.

Wie soll die Fakultätsfunktion programmiert werden?

'But things change drastically as soon as we start talking about an implemented programming language.' E.W. Dijkstra, 1988

Die Fakultätsfunktion dient in Lehrbüchern über Programmieren oft als Beispiel für die Rekursion. Dabei findet man oft Formulierungen wie etwa, frei ausgedrückt, die folgende:

```
function Fak1 (n: Integer): Integer;
begin if n=0
      then return 1
      else return n* Fak1 (n-1) end;
end Fak1;
```

Fak1 übersetzt die übliche mathematische Definition

$$0! = 1,\ n! = n*(n-1)!\ \text{für}\ n > 0$$

direkt in ein Programm und sieht trügerisch einfach aus. Bei genauer Betrachtung fällt aber ein wichtiger Unterschied auf: Fak1 verwendet 'Integer' als Argumenttyp und als Resultattyp. Dies geht wohl auf Algol 60 zurück, das weithin zur Publikation von Algorithmen verwendet wurde und noch keine Bereichstypen kannte. Die mathematische Definition spricht aber von positiven bzw. nichtnegativen ganzen Zahlen. Dieser Diskrepanz wegen scheitert Fak1 sowohl theoretisch wie auch praktisch, sobald es mit einer negativen ganzen Zahl aufgerufen wird.

Also flicken wir diese Funktion: auf altmodische Art, indem wir für die nicht definierten Werte etwas Vernünftiges einsetzen, z.B.: if $n \leq 0$ then return 1, oder indem wir für alle vorkommenden Wertebereiche Bereichstypen einführen:

```
function Fak2 (n: Cardinal): Natural;
```

Aber leider krankt dieses ‚verbesserte' Fak2 immer noch am selben Problem wie Fak1. 'Natural' entspricht nicht den natürlichen Zahlen, ebensowenig wie 'Cardinal' den nichtnegativen. Der Unterschied wird bereits

ersichtlich, wenn wir Fak 2 (13) auf einem 32-Bit-Rechner aufrufen. 13! = 6 227 020 800 kann nicht dargestellt werden in einem Zahlensystem, in dem $2^{31}-1 = 2\,147\,483\,647$ die größte ganze Zahl ist.

Flicken wir Fak 2 nach demselben Rezept. Mit der Definition:

```
type FakArgTyp = 0..12;
```

wird

```
function Fak 3 (n: FakArgTyp): Natural;
```

wohl korrekt? Zumindest auf geeigneten Computern?

Falls ja, können wir jetzt eine erste Antwort auf die Frage im Titel geben: Diese Fakultätsfunktion mit gestutzten Flügeln soll als Wertetabelle programmiert werden:

```
Fak 4: array (FakArgTyp) of Natural:=
                ( 0=>1,
                  1=>1,
                  2=>2,
                  3=>6,
                  4=>24,
                  5=>120,
                  6=>720,
                  7=>5_040,
                  8=>40_320,
                  9=>362_880,
                 10=>3_628_800,
                 11=>39_916_800,
                 12=>479_001_600);
```

Fak 4 ist klar und übersichtlich, und Laufzeitmessungen zeigen, daß diese Funktionstabelle 20- bis 50-mal schneller läuft als iterative und rekursive Programme für die Fakultät.

Warum findet man diese klar überlegene Lösung kaum in Lehrbüchern?? Weil niemand (mit Ausnahme von Schülern in Programmierkursen) die Fakultätsfunktion je wirklich programmieren muß und es daher gar nicht darauf ankommt, wie sie programmiert wird. Zwar tritt 'n!' in beliebig vielen mathematischen Formeln auf, bei deren Auswertung aber wird n! nur in Kombination mit anderen Größen berechnet. Betrachte als Beispiel die Reihenentwicklung: $e^x = 1 + x + x^2/2! + x^3/3! + ..$
Auch ohne Kenntnisse der Numerik drängt sich hier auf, die explizite Auswertung von n! durch eine iterative Berechnung der Terme zu ersetzen. Eine Partialsumme S, ein Term T, und ein Index i werden initialisiert: S:=1+x; T:=x; i:=2; dann steigt man in die Schleife ein:

```
while abs (T) > epsilon
do T:=T*x/i; S:=S+T; i:=i+1 end;
```

Als zweite, treffendere Antwort auf die Frage im Titel schlagen wir daher vor: *Gar nicht!* Wir würden aber gerne von Lesern hören, die einen triftigen Grund für das Programmieren der Fakultätsfunktion angeben können.

Als didaktische Herausforderung an unsere Leser, besonders an die Autoren und Lehrer, fragen wir: Kennt die Informatik keine besseren Beispiele der Rekursion, welche sich zur Einführung ins Programmieren eignen? Zeigen Sie uns *wenig bekannte,* einfache, kurze, prägnante Beispiele, in denen die Rekursion sich als natürlich und effizient erweist.

J. F. H. Winkler
Siemens AG ZFE F2 SOF3 Ref
Otto-Hahn-Ring 6
8000 München 83

J. Nievergelt
ETH Zürich
Institut für Informatik
CH-8092 Zürich

Neue Bücher

Informatik Spektrum

Tagungsberichte/Sammelbände

Meuer, H. W. (Hrsg.): **Supercomputer '89.** Anwendungen, Architekturen, Trends. (Informatik-Fachberichte 211) 1989. VIII, 171 S., brosch. DM 35,-. Berlin-Heidelberg-New York: Springer ISBN 3-540-51310-8

Wie jedes Jahr seit 1986 traf sich in Mannheim auf Einladung des Vereins zur Förderung der wissenschaftlichen Weiterbildung an der Universität Mannheim e. V. die deutschsprachige Supercomputergemeinde - Anwender, Betreiber, Hersteller - zu einem Dialog und Erfahrungsaustausch. In dem diesjährigen Seminar „Supercomputer - Anwendungen, Architekturen und Trends" wurden die neuesten Entwicklungen dieses stark innovativen Gebiets unter einem sehr anwendungsbezogenen, praktischen Aspekt aufgearbeitet. Experten aus dem Inland sowie den nach wie vor führenden Ländern USA und Japan konnten auch für diese zum vierten Mal stattfindende Veranstaltung zu Präsentationen gewonnen werden. Alle Beiträge zum Seminar sind in diesem Band enthalten. Insbesondere befassen sie sich mit folgenden Schwerpunkten:

- Visualisierung/Mensch-Maschine-Schnittstelle
- Architekturen
- Parallel Versus Vector Processing
- Innovative Anwendungen.

Fuhrmann, S., Pietsch, Th. (Hrsg.): **Praktische Anwendungen moderner Bürotechnologien.** 1989. 207 S., kart. DM 49,-. Berlin: Schmidt. ISBN 3-503-02809-9

Auf der achten Fachtagung des Fachausschusses 5.6 „Büroinformations- und Kommunikationssysteme (BIKOS)" der Gesellschaft für Informatik (GI) wurde Gelegenheit gegeben, über Erfahrungen bei Aufbau und Anwendung moderner Bürotechnologien zu berichten und auftretende Probleme und Wege zu deren Lösung zu diskutieren. Die Erfahrungsberichte der Tagungsreferenten sind im vorliegenden Band zusammengefaßt. Er versteht sich als Beitrag zur Schaffung des „Büros der Zukunft", das sich nicht an euphorischen Versprechungen der Hersteller neuer Bürotechnik und der Faszination am technisch Machbaren orientiert, sondern an gewachsenen