# Wieviel Programmierkenntnisse braucht ein Mathematiklehrer im Zeitalter von Computeralgebra?

**Karl Josef FUCHS, Salzburg**

Zunächst möchte in voranschicken, dass ich mit diesem Vortrag keine Antwort und schon gar kein Kochrezept anbieten möchte, wie man mit dem Phänomen Programmieren im Mathematikunterricht im Zeitalter von CAS umgehen sollte bzw. umzugehen hätte.

## 0. Woher die Motivation zu dieser Fragestellung?

- Jahrelang - etwa seit Mitte der 80er Jahre - haben vor allem Mathematiklehrer den Unterricht in Informatik getragen. Nach und nach hat daher die Ausbildung im Lehramt Mathematik den speziellen Anforderungen an einen Informatiklehrer Rechnung getragen. Die Ausbildung umfasste dabei auch die Beherrschung mindestens einer höheren Programmiersprache. Jetzt seit der Einführung eines Lehramtes Informatik und Informatikmanagement an den Universitäten Wien, Klagenfurt und Salzburg wird man wohl wieder auf diese veränderte Situation reagieren müssen.
- Viele von uns können sich sicherlich auch noch an die Diskussion nach der adäquaten, altersgemäßen Programmiersprache für die Einführung des Computers in den naturwissenschaftlichen Unterricht erinnern, die vor allem die Mathematik-Didaktik in den 80er Jahren bis hinein in die 90er Jahre beherrschte. Besonders heftige Diskussionen hat dabei das Buch *Mindstorms - Kinder, Computer und Neues Lernen* von Seymour Papert ausgelöst [PAPERT, 1982/ BENDER, 1987 uva]. Je stärker aber schließlich die Computeralgebra in den Mittelpunkt mathematik-didaktischer Fragestellungen rückte, desto mehr verschwanden Beiträge über den Computer als Programmierwerkzeug aus den einschlägigen Journalen.

Dennoch erhebt sich für mich die Frage: Wurde damit die Diskussion über das Programmieren im Mathematikunterricht tatsächlich beendet oder nur verdrängt?

- Geht man davon aus, dass man in den gängigen Computeralgebrasystemen DERIVE™, MAPLE™ und MATHEMATICA™ mit den vordefinierten Funktionen das Auslangen finden wird, so wird man aber bald bemerken, wie oft man bei individuellen Problemlösungen - in Form selbst kleinster Module [ASPETSBERGER/ FUCHS, 1995]- auf das Verketten vorgegebener Funktionen angewiesen ist. Seit einigen Jahren haben wir nun bereits Computeralgebra im Taschenrechnerformat im Mathematikunterricht zur Verfügung. Nicht zuletzt stellen die `Symbolrechner TI-92` von Texas Instruments mit dem `Program-Editor` oder der Symbolrechner Cassiopeia mit dem CAS MAPLE™ Elemente einer imperativen Programmiersprache zur Verfügung. Müssen wir nicht auch darauf reagieren, wenn wir den angehenden Lehrern umfassende Kenntnisse im Umgang mit diesen neuen Werkzeugen vermitteln wollen? Welche Inhalte sind dabei zu vermitteln? Gibt es so etwas wie fundamentale Prinzipien, die trotz rasch ändernder Technologie Bestand haben? Dieser Fragestellung wird im Folgenden unser besonderes Interesse gelten.
- Nicht zuletzt gibt bzw. gab es für mich ein pragmatisches Argument, mich mit der Frage des Programmierens im Zeitalter von Computeralgebra zu beschäftigten, da ich bereits vor einem Jahr mit der Leitung einer Special Group zum Thema *Programming in the Age of CAS* bei der ICTMT 5 an der Universität Klagenfurt betraut wurde und mich diese Frage somit seit einigen Monaten beschäftigt. Weiters fand am 11. Jänner 2001 ein Diskussionsabend mit AHS- und BHS-Lehrern zum Thema *Wieviel Programmierkenntnisse braucht ein Mathematiklehrer?* an der Universität Innsbruck - beeinflusst durch die Neuorganisation von Inhalten im Studienplan Mathematik - Lehramt - statt, bei dem ich zu einem Einführungsvortrag eingeladen wurde.

## 1. Programmieren - Schattierungen und Akzente

Besonders lohnenswert war es für mich - obwohl ich selbst seit Beginn des Informatikunterrichts dieses Fach an Gymnasien unterrichtet habe und noch unterrichte - für diesen Vortrag nach Definitionen des Begriffs Programmieren in Lehrbüchern der Didaktik der Mathematik und in Lehrbüchern der Didaktik der Informatik zu suchen.
Allerdings Fragen, wie

> Programmierphilosophien, Strukturiertes Programmieren,
> Strategische Vorgangsweise beim Programmieren (Kodieren - Editieren-Debuggen)

bleiben in diesem Beitrag ausgeblendet. Nicht deswegen, weil ich eine didaktische Diskussion für nicht erstrebenswert oder unbedeutend halte, sondern weil die Behandlung auch dieser Fragestellungen den Rahmen des Vortrags sprengen würde.

Jochen Ziegenbalg unterscheidet in seinem Didaktik -Lehrbuch *Algorithmen* u. a. zwischen

> *Imperativem Programmieren: Erstellen einer Folge von Befehlen, durch die der Zustand des Computers verändert wird (i. a. durch die Veränderung der Speicherbelegung),*
> *Funktionalem Programmieren: Kodiert werden Funktionen, die einen Satz von Eingabewerten (Argumente) in einen Satz von Ausgabewerten (Funktionswerte) transformieren.*

[ZIEGENBALG, 1996].

Im Lehrbuch *Informatik für den Lehrer* von Hans W. Haas und Detlef Wildenberg finden wir in *Band 2: Komplexere Probleme und Didaktik der Schulinformatik* im Abschnitt Programmieren:

> *Das Ziel ist die Realisierung von Algorithmen (und Datenstrukturen) auf einem Rechner.* [HAAS/ WILDENBERG, 1982]

Peter Hubwieser schließlich sieht in seiner *Didaktik der Informatik* die

*Programmierung als Implementierung von abstrakten Modellen, mit dem Ziel, diese durch Simulation zu veranschaulichen und zu überprüfen.* [HUBWIESER 2000]

Ähnlich sieht es der Wiener Informatikprofessor Gerald Futschek in seinem Beitrag *Informatik als Wissenschaft* in [REITER/ RIEDER, 1990]:

> *Zur Lösung eines Anwenderproblems entwirft der Informatiker zunächst ein Modell der Anwendung ... Das erste Modell wird in eine Reihe neuer Modelle umgeformt, die immer genauer und formaler werden, bis ein ablauffähiges Modell in einer bestimmten Programmiersprache erreicht ist.* [FUTSCHEK, 1990]

Bei Gustav Pomberger schließlich finden wir unter *Kapitel 3 Prozedur-orientierte Programmierung* in [RECHENBERG/ POMBERGER, 1999] zum Begriff Programmieren:

> *Programmieren im Sinne der Informatik heißt, ein Lösungsverfahren für eine Aufgabe so zu formulieren, dass es von einem Computer ausgeführt werden kann.* [POMBERGER, 1999]

## 2. Gibt es zeitlose Ideen im Programmierunterricht?

Damit gelangen wir zum eigentlichen Kern meines Vortrags. Erinnern wir uns daran, dass ich eingangs gesagt habe, dass wir unser Augenmerk auf zeitlose fundamentale Ideen oder Prinzipien des Programmierunterrichts richten müssen, wenn wir der Frage des Umfangs an Programmierkenntnissen, die ein Mathematiklehrer im Zeitalter von CAS mitbringen soll, nachgehen wollen.

Werfen wir dazu noch einmal einen Blick auf die Liste der Begriffsbestimmungen des Programmierens. Bei der Durchsicht der unterschiedlichen Definitionen wurde mir erst deutlich, wie stark allgemein anerkannte fundamentale Ideen der Mathematik zur Begriffsbestimmung herangezogen werden.

So finden wir in den Begriffserklärungen zum Programmieren die Idee des Algorithmus oder Lösungsverfahrens.
Auch Hans-Christian Reichel sieht in seinen *Fundamentalen Ideen der Angewandten Mathematik* in der Betonung von Algorithmen „... eine der wohl bekanntesten Fundamentalen Ideen der Anwendungsorientierten Mathematik...." [REICHEL 1995]
Elmar Cohors-Fresenborg wiederum betont einen engen Zusammenhang zwischen Funktionsbegriff, Modellbildung und Algorithmus, wobei im Prozess der Modellbildung vom Schüler funktionale Zusammenhänge zwischen einzelnen Größen erkannt und als Algorithmen kodiert werden [COHORS-FRESENBORG 1987].
Man beachte, dass gerade im Prinzip der Implementierung von Modellen ein charakteristisches Merkmal des Programmierens gesehen wird [Definitionen HUBWIESER, 2000 und FUTSCHEK, 1990].

Dies gilt natürlich auch oder im Besonderen im Zusammenhang mit Computeralgebrasystemen

> von der einfachen Kodierung und Verkettung elementarer logischer Funktionen [FUCHS, 1998]

Kodierung mit DERIVE™

```
konj(a, b) :=
  If a = 1 ∧ b = 1
     1
     0

disj(a, b) :=
  If a = 0 ∧ b = 0
     0
     1

neg(a) :=
  If a = 0
     1
     0

impl(a, b) := disj(neg(a), b)
VECTOR(VECTOR(impl(i, j), i, 0, 1), j, 0, 1)
```

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

bis hin zur Darstellung eines Rückkoppelungsprozesses wie ihn Josef Lechner [LECHNER 1996] beschreibt:

Kodierung mit MATHEMATICA™

```
F[x_]:=x+0.3(20-x)

NestList[F,6,20]

{6,10.2,13.14,15.198,16.6386,17.647,18.3529,18.847,19.192
9,19.435,19.6045,19.7232,19.8062,19.8644,19.905,19.9335,1
9.9535,19.9674,19.9772,19.984,19.9888}
```

Hans-Joachim Vollrath nennt den Begriff des funktionalen Denkens (im Sinne des Funktionalen Programmierens in der Begriffsdefinition von Jochen Ziegenbalg) sogar einen bedeutenden „didaktischen Impuls" des Mathematikunterrichts [VOLLRATH 1989].

Offen ist nun noch die Idee der Datenstruktur wie wir sie in der Definition von Haas und Wildenberg finden. Es handelt sich dabei jedoch um eine Idee, die ich ebenfalls 1994 in einem Aufsatz zur *Didaktik der Informatik* als fundamental angesehen habe [FUCHS 1994].
Dass wir offenbar mit dieser Ansicht nicht allein sind, zeigt auch das neu bearbeitete Lehrbuch *Mathematik mit dem TI-92 und TI-92 Plus* von Hans-Christian Reichel und Robert Müller, wo wir bereits in einem Eingangskapitel eine Diskussion über *Mengen - Listen - und Intervalle* und deren Implementierung in ein Computeralgebrasystem finden, eine Diskussion, die dann im Kapitel über *Daten- und Beziehungsstrukturen* fortgesetzt wird [REICHEL/ MÜLLER, 2001].
Man gestatte mir an dieser Stelle eine Bemerkung: Gerade bei diesem Lehrplankapitel wird für mich wieder einmal deutlich, mit welch großem didaktischen Geschick und mit welcher Umsicht der österreichische Lehrplan für die Oberstufe in Mathematik unter der Federführung von Heinrich Bürger 1989 formuliert wurde. Nämlich flexibel genug, um den Einbau von aktuellen Fragestellungen über Daten- und Beziehungsstrukturen im Zusammenhang mit Computeralgebrasystemen zu ermöglichen. [BÜRGER 1989]

Wollen wir also unseren angehenden Mathematiklehrern einen mündigen Umgang mit dem neuen Werkzeug Computeralgebra ermöglichen, so sollten wir ihnen auch den Umgang mit Datenstrukturen nicht vorenthalten, eine Fähigkeit, die unbestritten jeder gute Programmierer beherrschen muss.

Eine Idee, nämlich die der bereits genannten Beziehungsstrukturen, hätte ich noch gerne für die Liste der Programmierkenntnisse eines Mathematiklehrers eingefordert, denn Problemlösungen für mathematische Aufgabenstellungen erfordern das Entwickeln von Lösungsstrategien und das ‚Aufspüren' von Zusammenhängen.
Auch Hans-Christian Reichel [REICHEL 1995] sieht im *Beschreiben von Lösungswegen / von Arbeitsabläufen* eine zentrale Idee der Angewandten Mathematik. Beim Beschreiben solcher Lösungswege werden Beziehungen in grafischen Form (Struktogramme / Flussdiagramme) dargestellt und genauso wie Hans-Christian Reichel meine ich, dass dieses Darstellen von Beziehungsstrukturen wesentlich zur Verfügbarkeit vorhandenen Wissens beim Schüler beitragen wird.

Im Mittelpunkt dieser Beziehungsstrukturen werden die Kontrollstrukturen

der Sequenz,
der Wiederholung und
der Verzweigung

stehen, wie sie in allen Computeralgebrasystemen implementiert sind.

Kodierung des „Klassikers Turmrechnung" mit dem (imperativen) Programmiermodul in MAPLE™

```
turm := proc(a)
        local ergebnis;
            ergebnis := a;
            for i to 2 do
                for j from 2 to 9 do
                    print(ergebnis);
                    if i = 1 then ergebnis := ergebnis*j else ergebnis := ergebnis/j fi
                od
            od
        end
```

17
34
102
408
2040
12240
85680
685440
6168960
3084480
1028160
257040
51408
8568
1224
153
17

## 3. Kurze Zusammenfassung

Die Antwort auf die eingangs gestellte Frage „Wieviel Programmierkenntnisse braucht ein Mathematiklehrer im Zeitalter von CAS?“ ist für mich - nach Abfassung dieses Beitrags für den Lehrerfortbildungstag 2001- vor allem in der Weise zu beantworten: Selbst dann, wenn man sich bei der Ausbildung des Mathematiklehrers dafür entscheidet, auf die Vermittlung einer höheren Programmiersprache zu verzichten, so bleiben dennoch eine große Zahl der als wesentlich angesehenen Definitionsmerkmale des Programmierens im Zeitalter der Computeralgebra als fundamentale Prinzipien des Mathematikunterrichts wirksam und diese gilt es sehr wohl bei der Ausbildung zu berücksichtigen.

**Literatur:**

ASPETSBERGER, Klaus; FUCHS, Karl (1995): *DERIVE im Mathematikunterricht: Zur Organisation von Beobachtungseinheiten; Modultechnik im Mathematikunterricht mit Computeralgebra.* In: Beiträge zum Mathematikunterricht, Franzbecker, Hildesheim, S. 74-77

BENDER, Peter (1987): *Kritik der LOGO-Philosophie.* In: JMD 8 (87) 1/2, S. 3-103

BÜRGER, Heinrich (1989): *Tendenzen in neuen österreichischen Mathematiklehrplänen.* In: Beiträge zum Mathematikunterricht, Verlag Franzbecker, S. 101-104

COHORS-FRESENBORG, Elmar (1987): *Zur Integration algorithmischer und axiomatischer Denkweisen in den Mathematikunterricht der Klasse 7 des Gymnasiums.* In: Beiträge zum Mathematikunterricht, Verlag Franzbecker, S. 130-133

FUCHS, Karl Josef (1994): *Didaktik der Informatik: Die Logik fundamentaler Ideen.* In: Medien und Schulpraxis 4+5, S. 42-45

FUCHS, Karl Josef (1998): *Computeralgebra - Neue Perspektiven im Mathematikunterricht.* Habilitationsschrift Universität Salzburg

FUTSCHEK, Gerald (1990): *Informatik als Wissenschaft.* In: REITER, Anton; RIEDER, Albert: Didaktik der Informatik, Jugend und Volk, Wien, S. 2-6

HAAS, Hans; WILDENBERG, Detlef (1982): *Informatik für Lehrer - Band 2: Komplexere Probleme und Didaktik der Schulinformatik.* Oldenbourg Verlag, München, Wien

HUBWIESER, Peter (2000): *Didaktik der Informatik.* Springer-Verlag, Berlin, Heidelberg, New York

LECHNER, Josef (1996): *Iteration und Rekursion - Problembeschreibung und Lösungsmethode zugleich.* In: Teaching Mathematics with DERIVE and the TI-92, ZKL-Texte Nr. 2, Münster, S. 303- 318

PAPERT, Seymour (1982): *Mindstorms - Kinder, Computer und Neues Lernen.* Birkhäuser Verlag

POMBERGER, Gustav (1999): *Prozedurorientierte Programmierung.* In: RECHENBERG, Peter; POMBERGER, Gustav: Informatik-Handbuch, Carl Hanser Verlag, München, Wien, S. 517-528

REICHEL, Hans-Christian (1995): *Fundamentale Ideen der Angewandten Mathematik.* In: Wissenschaftliche Nachrichten, S. 20-25

REICHEL, Hans-Christian; MÜLLER Robert (2001): *Mathematik mit dem TI-92 und dem TI-92 Plus.* Verlag öbv & hpt, Wien

ZIEGENBALG, Jochen (1996): *Algorithmen.* Texte zur Didaktik der Mathematik, Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford

**Anschrift des Verfassers:**

Univ.Doz. Prof. Mag. Dr. Karl Josef FUCHS
Universität Salzburg
Gastprofessor für Didaktik der Mathematik, Universität Innsbruck

Hellbrunnerstraße 34
5020 SALZBURG
karl.fuchs@sbg.ac.at