Karl Josef FUCHS & Simon PLANGG, Salzburg

Programmieren mit Hand Held Technologie

Bedeutung und Motiv zum Programmieren im Mathematikunterricht

2005 hat Hans-Stefan Siller mit seiner Dissertation Modellbilden-Eine zentrale Leitidee der Mathematik (2008) auf die besondere Bedeutung der Modellbildung als Lehr- und Lernziel eines modernen Mathematikunterrichts hingewiesen. Mit dem Implementieren von mathematisierten Modellen im Rahmen einer Modellbildung rückt neben dem Operieren mit Technologie in der ‚Welt der Mathematik' nun auch die Tätigkeit des Programmierens wieder in den Fokus methodisch-didaktischer Betrachtungen. Deutlich wird diese Erweiterung des Konzepts aus der Mathematik von Dominik Leiß und Werner Blum (2006) durch die Hinzunahme einer ‚Welt der Informatik' im Modellierungskreislauf von Hans-Stefan Siller (Siller 2009, S. 405).
Für den Schritt der Implementierung, d.h. für das *Programmieren*, verwenden wir in unserem Beitrag den Hand Held TI-Nspire. Für die Verwendung dieser Hardware spricht für uns ─ neben zahlreichen weiteren Argumenten für die Hand Held Technologie (Fuchs & Plangg 2018, S.7-9) ─ die *leichte Handhabbarkeit/Portabilität* der Technologie für den Einsatz im Unterricht. Wir weisen jedoch darauf hin, dass wir für die Implementierung den Editor im Computer Algebra System (CAS) des Rechners verwenden, obwohl keinerlei spezifisch symbolische Fähigkeiten des CAS für die Lösung der nachfolgenden ‚numerischen' Aufgabe genutzt werden. Für die Implementierung könnte daher z. B. auch der Grafikrechner der TI-Nspire-Produktlinie verwendet werden.

Prototypische Vorgangsweise beim Programmieren mit Technologie

Wir wollen nun darstellen, dass *Programmieren* (d.h. Implementieren von *‚lauffähigen' Modellen* (Futschek 1990)) ein mehrschrittiger Prozess ist, der nicht nur aus einem ‚bloßen' Kodieren in einer Programmiersprache besteht. Folgende Schritte sind für uns für einen Implementierungsvorgang unbedingt notwendig (Fuchs & Plangg 2018, S.22):

- Prototypische Problemstellung
- Wahl des Programmierparadigmas
- Einzelschritte Algorithmischen Denkens
  - Analyse und Präzisierung des Problems

- Grafische Darstellung des Lösungsalgorithmus (Struktogramme/ PROGRAPH-Diagramme)
- Implementierung eines Lösungsalgorithmus
- Endlichkeits-, Effektivitätskontrolle, (allenfalls) Modifikation/Adaption des Algorithmus sowie Bewertung der Ergebnisse im Lichte der Realität.

Näherungsweise Berechnung von Wurzeln

Bereits Karl Josef Fuchs hat das HERONsche Wurzelziehen als prototypische Problemstellung für eine numerisch-handelnde Vorgehensweise im Mathematikunterricht mit CAS dargestellt (Fuchs 2001). Auch in diesem Beitrag soll anhand dieses Verfahrens ein Näherungswert für $\sqrt{5}$ berechnet werden, allerdings mit dem Fokus auf dessen Implementierung über den Programmeditor des TI-Nspire CX CAS. Für eine mögliche vorangestellte Aufbereitung einer geometrischen Interpretation des Heron-Verfahrens siehe bspw. Elschenbroich & Seebach (2015). Ausgangspunkt der Problemstellung könnte dabei die Frage sein, wie der Taschenrechner Näherungswerte für Wurzeln bestimmt.

Vor der eigentlichen Implementierung dieser Problemstellung gilt unser Augenmerk zunächst noch der Wahl des Programmierparadigmas (*Structured Programming im prozeduralen Paradigma* (Dijkstra, Hoare & Dahl 1972 oder *Funktionales Paradigma* (Hubwieser 2007, S.147-152; 201-207)). Nach dem *Prozeduralen Paradigma* besteht ein Programm aus einer Abfolge von Anweisungen, insbesondere aus Kontrollstrukturen wie Sequenz, Verzweigung und Wiederholung. Die Sprachelemente zur Implementierung von Algorithmen nach dem *Funktionalen Paradigma* hingegen sind Funktionen in durchaus strengem mathematischem Sinn (ebd.). Die Ausführung von Programmen nach diesem Paradigma besteht dann im Wesentlichen aus einer Verkettung einzelner Funktionen. In Hinblick auf die in diesem Beitrag eingesetzte Technologie können Programme (*Prozedurales Paradigma*) lediglich in einer Calculator-Ansicht aufgerufen werden, ermöglichen jedoch beispielsweise über die Verwendung von Abfragen (Abb. 2, links) einen anwenderfreundlichen Einsatz. Im Gegensatz dazu können erstellte Funktionen (*Funktionales Paradigma*) zudem sowohl in der Grafik- als auch in der Spreadsheet-Ansicht aufgerufen werden und ermöglichen so eine flexiblere Nutzung als beispielsweise analoge Programme. Hingegen gilt es beim Einsatz von Funktionen über die entsprechenden Argumente der Funktionen, z. B. deren Anzahl und Reihenfolge (Abb. 2, rechts), Bescheid zu wissen. Selbst erstellte Hilfen bzw. Kommentare können dabei nützlich sein. Ein Abspeichern und Aufrufen der so erstellten Programme bzw. Funktionen in

einer Bibliothek ermöglicht ein modulorientiertes Arbeiten im Unterricht (vgl. Aspetsberger & Fuchs, 1995).
Für die Darstellung eines Algorithmus sind, in Abhängigkeit des Fortschritts der Lernenden, verschiedene Ausprägungen denkbar. Diese umfassen eine verbale Beschreibung des Algorithmus, bspw. durch Angabe einer Protokoll-Liste sowie eine graphische Darstellung des Algorithmus, bspw. anhand eines Struktogramms (Abb. 1, links) bzw. eines PROGRAPH-Diagramms (Abb. 1, rechts). Für den Anfang bietet sich vor allem die verbale Beschreibung an. Eine Vorbereitung auf das Erstellen eines Programms kann dann durch die Erstellung einer graphischen bzw. formal-abstrakten Darstellung unterstützt werden.

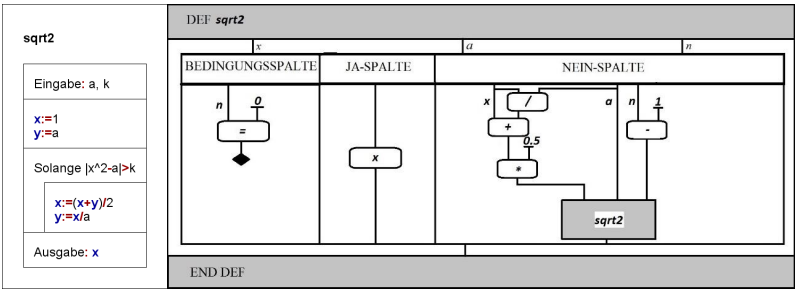

Abb. 1: Struktogramm (links) und PROGRAPH-Diagramm (rechts) des Heron-Verfahrens

Darauf aufbauend ist eine Implementierung des Heron-Verfahrens in den Programmeditor des TI-Nspire CX CAS sowohl nach dem *Prozeduralen* als auch dem *Funktionalen Paradigma* (Abb. 2) denkbar.

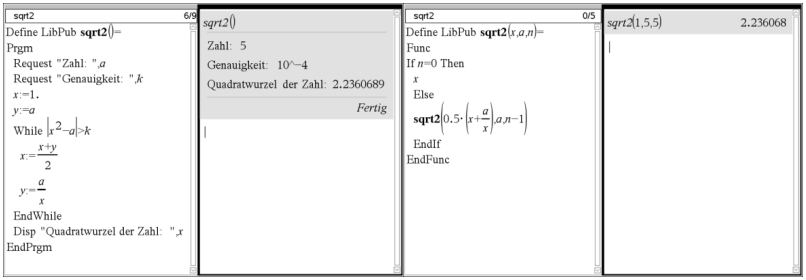

Abb. 2: Implementierung des Heron-Verfahrens in den Programmeditor des TI-Nspire CX CAS nach dem Prozeduralen (links) sowie Funktionalen Paradigmas (rechts)

In der Phase der anschließenden Analyse und Modifikation des Algorithmus erscheinen mehrere Vorgehensweisen möglich. In einem ersten Schritt könnten in Bezug auf das Programm die für die angegebene Genauigkeit notwendige Anzahl von Iterationen ausgegeben werden. Dies kann über die

Einführung einer Zählvariable umgesetzt werden. Eine mögliche Feststellung dabei könnte sein, dass für das Erreichen der Maschinengenauigkeit des TI-Nspire CX CAS nicht mehr als sechs Iterationen notwendig sind. Eine weitere Möglichkeit besteht auch darin, die implementierten Algorithmen auf $\sqrt[3]{5}$, $\sqrt[4]{5}$, … $\sqrt[n]{5}$ zu verallgemeinern und dabei auf die Probleme bei der Verallgemeinerung hin zu weisen (vgl. Fuchs 2001).

Umsetzung in der Lehre

Die Aufgabe HERONsches Wurzelziehen sowie die daran anschließenden Weiterentwicklungen gemäß des Algorithmischen Denkens waren Inhalte der Pflichtlehrveranstaltungen Computeralgebra im Mathematikunterricht (mehrfach im auslaufenden Diplomstudium und BA Studium 2013) sowie Technologie I (im neuen Cluster BA-MA Studium 2016) für Lehramtskandidat(inn)en an der Universität Salzburg.

Literatur

Aspetsberger, K., Fuchs, K. J. (1995). Derive im Mathematikunterricht: Zur Organisation von Beobachtungsfenstern, Modultechnik im Mathematikunterricht mit Computeralgebra. In: *Beiträge zum Mathematikunterricht*, Bad Salzdetfurth ü. Hildesheim: Franzbecker Verlag, S. 74-77.

Dijkstra, E. W., Hoare, T. & Dahl, O-J. (1972). *Structured Programming*. London: Academic Press Ltd.

Elschenbroich, H-J. & Seebach, G. (2015). Das Heron-Verfahren geometrisch betrachtet. Mathematik lehren, Heft 188, S. 30-33.

Fuchs, K. J. (2001). Computer algebra systems in mathematics education-Teacher Training programs, challenges and new aims. *Zentralblatt für Didaktik der Mathematik*, Jg. 35, Heft 1, S.20-23.

Fuchs, K. J. & Plangg, S. (2018). *Computer Algebra Systeme in der Lehrer(innen)*bildung. Münster: WTM Verlag.

Futschek, G. 1990). Informatik als Wissenschaft. In: *Didaktik der Informatik* (Reiter, A. Hrsg.), Wien: Jugend und Volk, S.2-6.

Hubwieser, P. (2007). *Didaktik der Informatik*. Berlin, Heidelberg: Springer Verlag.

Leiß, D. & Blum, W. (2006). Beschreibung zentraler mathematischer Kompetenzen. In:
Bildungsstandards Mathematik; konkret (Blum, W. et al Hrsg.), Berlin: Cornelsen Scriptor, S. 33-50.

Siller, H-S. (2008). Modellbilden-Eine zentrale Leitidee der Mathematik. In: *Schriften zur Didaktik der Mathematik und Informatik an der Universität Salzburg-Band 2* (Fuchs, K. J. Hrsg.), Aachen: Shaker Verlag.

Siller, H-S. (2009). Der Begriff „Modellbilden“ in der Mathematik- bzw. Informatikdidaktik. In: *Beiträge zum Mathematikunterricht*, Münster: WTM Verlag, S.403-406.