MODUL 2: WIE DENKEN COMPUTER?

HINWEISE FÜR LEHRPERSONEN DER SEKUNDARSTUFE II (DEZEMBER 2008)

DIDAKTISCHE ÜBERLEGUNGEN

Die Informationsgesellschaft wird geprägt durch Computer. Auch wenn zu deren Nutzung Programmierung meist nicht notwendig ist, gibt es trotzdem wichtige Gründe, warum ein Grundverständnis des Programmierens zur Allgemeinbildung gehören sollte.

Eine erste Argumentationslinie nimmt direkt Bezug auf die immer grössere Präsenz von programmierten Maschinen in unserer Umwelt:

- **Förderung der Nutzungskompetenz:** Das Verständnis der grundlegenden Strukturen und Abläufe in informationsverarbeitenden Systemen fördert deren effektive und effiziente Nutzung.

- **Förderung der Entscheidungskompetenz:** In demokratischen Strukturen wie der Schweiz sollen mündige Bürgerinnen und Bürger kompetent über Sachfragen entscheiden. In der Informationsgesellschaft betrifft dies in zunehmendem Masse auch Themen der Informatik.
- **Förderung der Gestaltungskompetenz:** Informationsverarbeitende Systeme sind nicht starr vorgegeben, sondern werden von Menschen entwickelt. Anwenderinnen und Anwender von informationsverarbeitenden Systemen sollten bei der Gestaltung dieser Systeme kompetent mitreden können, da diese Systeme ihre Arbeitsabläufe wesentlich prägen. Dies erfordert nicht nur entsprechendes Know-how, sondern auch die Einsicht, dass informationsverarbeitende Systeme überhaupt gestaltbar sind.

Die zweite Argumentationslinie steht unter dem Motto *"Modellieren / Programmieren ist ein vielseitiges Denkwerkzeug"*.

- **Denken in Modellen:** Das Modellieren / Programmieren fördert das Denken in Modellen und bietet für verschiedene Problemklassen brauchbare Werkzeuge zu deren Lösung.
- **Analytische Schärfe:** Zudem fördert Programmieren/Modellieren die analytische Schärfe, da Programmieren ein detailliertes Beschreiben von Prozessen erfordert, welches keine Zweideutigkeit mehr zulässt.
- **Nachweis des Verständnisses von Prozessen:** Um einen Prozess modellieren bzw. programmieren zu können, muss er ganz verstanden werden. Programmieren eignet sich somit als Nachweis des Verständnisses eines Ablaufs
- **Ausprobieren und Nachvollziehen durch Simulation:** Programmieren ermöglicht durch Simulation auch das Ausprobieren und Nachvollziehen verschiedener Varianten. Dies wiederum fördert die Auseinandersetzung mit den modellierten/programmierten Sachverhalten.

LEKTIONSIDEEN

- Lektion 1: Einführung ohne Computer
- Lektion 2: Videotutorial
- Lektion 3: Selber programmieren
- Lektion 4: Eigenes Projekt

LEKTION 1: EINFÜHRUNG

Menschen denken anders als Computer. Computer befolgen nämlich stur ihre Regeln. Menschen haben damit ihre liebe Mühe....
Die Lektion findet komplett ohne Computer statt. Damit soll einerseits die Erwartungshaltung *"Informatik = Computer"* gebrochen werden, andererseits soll diese enaktive Erfahrung in Erinnerung bleiben.

T	Was	Sozial-form	Medien
5	Einführung und Story	Plenum	-
10	Lektion ohne Computer (Situation 1)	4 Gruppen	Klebeband, Stühle, Steuerungszeichen, Leeres Zimmer
5	Feedbackrunde zu Situation 1	Plenum	Programme der Gruppen
5	Situation 2	4 Gruppen	s. Situation 1
5	Feedbackrunde	Plenum	s. Situation 1
5	Situation 3	4 Gruppen	s. Situation 1
5	Feedbackrunde	Plenum	s. Situation 1

Hinweis: Diese Lektion wurde im Rahmen des Schweizer Jahres der Informatik am Tag der Informatik mehrfach erprobt. Entsprechende Verbesserungen aufgrund der gemachten Erfahrungen sind in dieses Modul eingeflossen. Ein Video auf iLearnIT.ch zeigt Eindrücke aus einer Lektion mit einer Klasse der Sekundarstufe I.

LEKTION 2: VIDEO-TUTORIALS

Um die Programmierumgebung Scratch einzuführen, eignet sich z.B. ein Tutorial. Im Videotutorial von iLearnIT.ch wird anhand eines kleinen Spiels die Programmierumgebung eingeführt. Die Schüler/innen sehen in fünf Videosequenzen, wie das Programmierbeispiel schrittweise aufgebaut wird. Dieses Programm kann selbst nachprogrammiert und erweitert werden.

T	Was	Sozial-form	Medien
15	Videotutorial	Einzel- oder Partnerarbeit	Computer mit Internet und Audio
5	Feedbackrunde	Plenum	-
20	Umsetzen der Tutorialaufgabe	Partner-arbeit	Computer mit Scratch

LEKTION 3: SELBER PROGRAMMIEREN

Um die Programmierkenntnisse zu vertiefen, sind Übungen sinnvoll. Anhand kleiner Programmierbeispiele werden beim Programmieren zentrale Strukturen eingeübt. Die Programmieraufgaben können alle heruntergeladen und zur Lösung erweitert werden. Vorhandene Lösungsbeispiele zeigen, wie eine mögliche Lösung aussehen könnte. Dieser Ansatz der schrittweisen Erweiterung von Programmen erleichtert den Zugang zur abstrakten Welt der Computeralgorithmen. Zeitbedarf: 1-2 Lektionen.

LEKTION 4: EIGENS PROGRAMMIERPROJEKT

Wenn Schülerinnen und Schüler ein eigenes Programmierprojekt umsetzen können, münden die Arbeiten oft in kreativen Lösungen. Dabei sollen aber gelernte Strukturen und Regeln berücksichtigt werden. Dies könnte z.B. in einem kleinen Programmierwettbewerb realisiert werden, bei dem bestimmte Vorgaben erfüllt werden müssen. Zeitbedarf: 2-4 Lektionen.

MÖGLICHER ZUSATZ:

Scratch bietet bereits eine Visualisierung der Algorithmen an, indem Programme mit farbigen Blöcken geschrieben werden, die eine gewisse Ähnlichkeit zu Struktogrammen (auch Nassi-Shneiderman-Diagramme genannt) aufweisen. Andere mögliche Repräsentationsformen sind der Programmablaufplan (PAP) - auch bekannt als Flussdiagramm oder der endliche Automat bzw. das Zustandsübergangsdiagramm. Die Schülerinnen und Schüler sollen ihre Programme als PAP und endliche Automaten darstellen oder aus bestehenden Visualisierungen die Programme umsetzen.