Informatik vor, im und nach dem Schweizer Gymnasium

Komm, D.
ETH Zürich

DOI: 10.18420/ibis-02-01-09

Zusammenfassung

In diesem Beitrag wird die aktuelle Situation der Schulinformatik in der Deutschschweiz beschrieben. Im Zentrum stehen Unterrichtsbeispiele zum Aufbau von Kompetenzen, die über die gesamte Schulzeit gefördert werden können. Schließlich wagen wir einen Blick in eine Zukunft, in der die 2024 angestoßenen Prozesse zur Umsetzung eines Grundlagenfachs Informatik, welche die gesamte Schweiz betreffen, weitgehend abgeschlossen sind.

Einleitung

Eine (nicht die einzige!) der Hauptaufgaben des Gymnasiums ist es, Schülerinnen und Schüler auf ein Hochschulstudium vorzubereiten. Was aber sind die grundlegenden Kompetenzen, die benötigt werden, um sich erst einmal in der Hochschule der Zukunft und im Anschluss auf dem Arbeitsmarkt der Zukunft behaupten zu können? Oder vielleicht noch fundamentaler formuliert: Was ist das eigentlich für eine Welt, auf die Gymnasium und Hochschule vorbereiten sollen? In einer schnelllebigen Zeit, in der sich manch eine und einer von der Digitalisierung vor sich hergetrieben fühlt, ist die Beantwortung dieser Frage nicht ganz einfach. Einen Teil der zukünftigen Jobs heutiger Zehntklässlerinnen und -klässler gibt es noch gar nicht.

Unumstritten ist jedoch, dass Informatikgrundlagen in den Berufen der Zukunft einen immer größeren Stellenwert haben werden. Von der Ärztin bis zum Logistiker wird es immer wichtiger werden, zumindest das Grundvokabular von Informatikerinnen und Informatikern zu verstehen. Gerade in der Forschung werden die Teams immer interdisziplinärer und es ist schlichtweg unrealistisch, keinen Kontakt zur Informatik zu erwarten. Sollen Probleme aus der Praxis mit Hilfe der Informatik gelöst werden, müssen sie abstrakt dargestellt in die *Sprache der Informatik* übersetzt werden und können dann (innerhalb gewisser Grenzen) gelöst werden. Die Erkenntnisse aus dem abstrahierten Modell erlauben wiederum Rückschlüsse für das ursprüngliche Problem; siehe Abbildung 1.

So enthält beispielsweise der nun bereits etablierte Bachelor-Studiengang *Humanmedizin* der ETH Zürich gleich zwei Informatik-Vorlesungen, die einmal Grundlagen des Programmierens und einmal *Data Science* ins Zentrum stellen. Der Medizin-Studiengang ist dabei keine

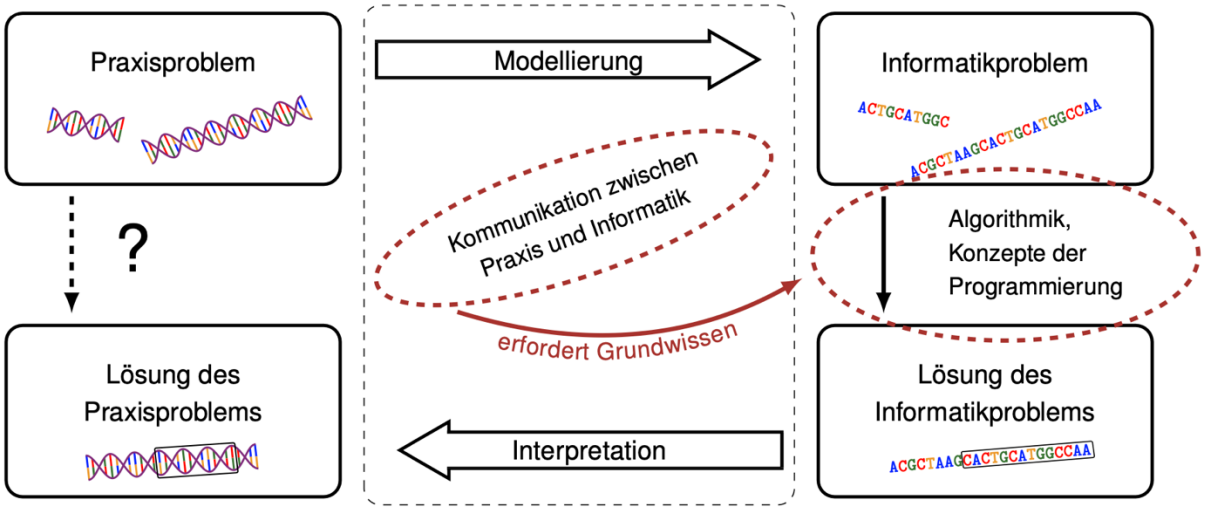

Abbildung 1: Modellierung als Informatikproblem und Interpretation der Lösung.

Ausnahme. Ganz im Gegenteil: Informatikkompetenzen werden in einigen Jahren zu den Grundlagen vieler weiterer Studiengänge gehören – wenn sie dies nicht bereits tun – und das aus gutem Grund; daneben seien natürlich auch per Definition interdisziplinäre Studiengänge wie Bioinformatik, Wirtschaftsinformatik etc. erwähnt, die sich großer Beliebtheit erfreuen.

Die Entwicklung der entsprechenden Lehrveranstaltungen stellt wiederum eine Herausforderung für die Dozierenden dar. Denn die Studierenden haben sich eben nicht für Informatik eingeschrieben und hatten in ihrer Schullaufbahn nicht unbedingt intensiven Kontakt mit der Disziplin; und aktuell bedeutet eine Grundlagenvorlesung somit tatsächlich, ganz am Anfang zu beginnen und nahezu kein Vorwissen vorauszusetzen.

Glücklicherweise nimmt die Informatik in der Schweiz aber auch in den Jahren vor dem Hochschulstudium – ja sogar in den Jahren vor der Sekundarstufe – einen immer wichtigeren Platz ein. Somit ist zu erwarten, dass die Studierenden der Zukunft bereits mit einem gefestigten Grundlagenwissen im Hörsaal sitzen werden und vieles, was aktuell noch an der Hochschule gelehrt wird, bereits aus der Schule bekannt sein dürfte.

In den folgenden Abschnitten wird ein teilweise ambitionierter – aber keineswegs unrealistischer – Blick in die Zukunft gewagt. Für die Schweizer Volksschule (diese umfasst den Kindergarten bis zu Klasse 6 (Ende der Primarstufe) bzw. Klasse 9 (Ende der Sekundarstufe I)) werden Informatikkompetenzen im *Lehrplan 21* formuliert,[1] der bereits vor einigen Jahren in den meisten deutsch- und mehrsprachigen Kantonen und Liechtenstein umgesetzt wurde; ähnliche Lehrpläne existieren für den französisch- und italienischsprachigen Teil des Landes.[2] Für das Gymnasium (je nachdem, wann der Übergang von der Volksschule stattfindet, handelt es sich entweder um ein *Lang-* oder *Kurzzeitgymnasium*) war die Informatik bislang ein sogenanntes

[1] https://www.lehrplan21.ch

[2] https://www.ciip.ch/Plans-detudes-romands/Plan-detudes-romand-scolarite-obligatoire-PER/Plan-detudes-romand-PER/

obligatorisches Fach, wird ab August 2024 (über einen Zeitraum von mehreren Jahren) allerdings zu einem *Grundlagenfach* aufgewertet, womit es genauso für die Schweizer Matura zählt wie beispielsweise Geschichte oder Biologie.

Informatik vor dem Gymnasium

Mit dem Schweizer Lehrplan 21 findet die Informatik offiziell Platz in der Schweizer Volksschule der 21 deutsch- und mehrsprachigen Kantone. Zwar ist nur ein kleines Stundengefäß vorgesehen, welches zu einem Teil zudem für den Aufbau von Anwendungs- und Medienkompetenzen verwendet werden soll, aber grundsätzlich formuliert der Lehrplan den begrüßenswerten Auftrag, Informatikkompetenzen bereits früh in der Schule aufzubauen.

Gerade in den ersten Jahren ihrer Schullaufbahn geht es darum, die Schülerinnen und Schüler zu begeistern und die Informatik (wahrheitsgemäß) vorzustellen als eine Disziplin, die sich zu einem Großteil mit dem kreativen Lösen von Problemen beschäftigt. Da im Lehrplan 21 vom Kindergarten bis zur vierten Klasse (je nach kantonaler Umsetzung) keine eigenen Lektionen für die Informatik vorgesehen sind, kann dies nur in anderen Fächern passieren.

Im Lehrplan 21 werden die aufzubauenden Informatikkompetenzen in *Datenstrukturen*, *Algorithmen* und *Informatiksysteme* unterteilt. Für diese werden dann wiederum Kompetenzstufen formuliert, die die Schritte des Aufbaus über die Schuljahre beschreiben. Im Folgenden werden drei Beispiele für Themen aufgezeigt, die in den unteren Schulstufen behandelt und mit denen unterschiedliche Kompetenzstufen abgedeckt werden können.

Darstellung von Daten in der Volksschule

Die Darstellung von Daten und alle Probleme, die sich daraus ergeben (*Geheimhaltung vor Dritten*, *Resistenz gegen Fehler*, *Effizienz der Darstellung*), machen einen wesentlichen Teil der Informatik aus. Im Lehrplan 21 findet sich unter anderem folgende Kompetenzstufe:

> **MI2.1.3g** Die Schülerinnen und Schüler verstehen die Funktionsweise von fehlererkennenden und -korrigierenden Codes.

Obwohl dies erst für die Sekundarstufe I vorgesehen ist, kann ein Grundverständnis bereits in den ersten Jahren der Volksschule aufgebaut werden. Wie auch bei vielen anderen Konzepten kann die Vermittlung in Form eines *Spiralcurriculums* umgesetzt werden. Dies bedeutet, dass fehlererkennende und -korrigierende Kodierungen in der Schullaufbahn immer wieder behandelt werden, wobei die Komplexität stetig gesteigert wird.

Begonnen werden kann spielerisch ganz vorne – und zwar mit Schülerinnen und Schülern, die womöglich noch nicht lesen, schreiben oder gar zählen können (Hauser et al., 2020):

1. Die Lehrerin einer Klasse ernennt eine Schülerin zu ihrer Assistentin und weiht sie in ein Geheimnis ein; vorgeführt wird ein Zaubertrick und die Assistentin wird fortan als *Hellseherin* bezeichnet. Unter Umständen findet eine Probe der beiden Protagonistinnen in einer vorangehenden Pause statt.
2. Danach steht die Lehrerin mit ihrer Klasse im Kreis um einen großen Tisch, auf dem sich ein Stapel Spielkarten befindet. Die Hellseherin verabschiedet sich für die nächsten Minuten und wartet vor der geschlossenen Türe des Schulzimmers.
3. Nun bittet die Lehrerin einen Schüler, einige Karten, die verdeckt auf dem Tisch liegen, aufzudecken. Hierzu gibt es keine genauen Vorgaben; zum Beispiel könnten insgesamt 13 Karten auf dem Tisch liegen, von denen nun acht aufgedeckt sind.
4. Als nächstes wird ein weiterer Schüler gebeten, entweder eine oder zwei beliebige Karten umzudrehen. Die Entscheidung liegt bei ihm.
5. Ist dies vollbracht, darf die Hellseherin den Raum wieder betreten; sie schaut kurz auf die Karten und kann im Anschluss angeben, ob eine oder zwei Karten vom zweiten Schüler umgedreht worden sind.

Dies kann beliebig oft wiederholt werden: Die Hellseherin verlässt den Raum, ein anderer Schüler deckt sechs Karten auf, eine weitere Schülerin dreht eine oder zwei um, die Hellseherin betritt den Raum und sieht sofort, was passiert ist.

Wie macht sie das? Nun, ein »kleines Detail« muss noch erwähnt werden. Bislang wurde davon ausgegangen, dass die erste Schülerin bzw. der erste Schüler jeweils eine gerade Anzahl Karten aufdeckt (im obigen Beispiel waren es acht bzw. sechs). Sollte es eine ungerade Anzahl sein, muss die Lehrerin eingreifen, damit der Zaubertrick funktioniert. In diesem Fall legt sie eine weitere aufgedeckte Karte dazu. Dies führt dazu, dass die Anzahl der aufgedeckten Karten immer gerade ist, bevor die zweite Schülerin bzw. der zweite Schüler eine oder zwei Karten umdreht. Und jetzt wird auch klar, was die Lehrerin und die Hellseherin als Strategie vereinbart haben: findet letztere eine ungerade Anzahl aufgedeckter Karten vor, wurde eine Karte umgedreht; findet sie eine gerade Anzahl vor, wurden zwei Karten umgedreht. Die Hellseherin »zählt« dazu im Kopf einfach alle aufgedeckten Karten durch: ungerade, gerade, ungerade, gerade, …

Das Konzept des *Prüfbits* kann noch expliziter gemacht werden, indem die Lehrerin auch im Falle einer geraden Anzahl aufgedeckter Karten eine weitere Karte dazulegt, aber diesmal eine verdeckte. Der Zaubertrick kann auch leicht so abgeändert werden, dass von der zweiten Schülerin bzw. vom zweiten Schüler entweder eine oder keine Karte umgedreht wird. Dies entspricht dann den beiden Situationen, dass ein Fehler passiert ist oder nicht. Bei der Übertragung einer Sequenz von Bits wird zum Beispiel versehentlich aus einer Null eine Eins.

Die Idee hinter den *fehlererkennenden Codes* kann in einem späteren Schuljahr zu einem *fehlerkorrigierenden Code* erweitert werden, mit welchem nicht nur erkannt werden kann, dass ein Fehler passiert ist, sondern auch, wo ein Fehler passiert ist. Mit diesem Wissen kann der Fehler dann wiederum automatisch korrigiert werden, was eine Grundlage moderner Datenübertragung darstellt. Die Idee einer Umsetzung mit Karten ist in Abbildung 2 dargestellt; den

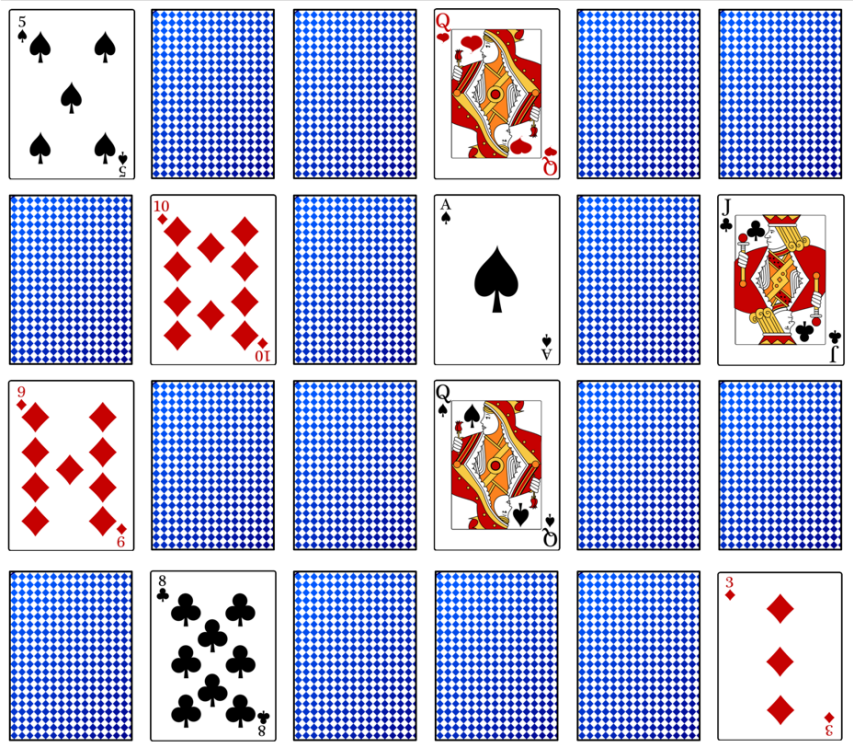

Abbildung 2: Ein fehlerkorrigierender Code. Zu Beginn war die Anzahl der aufgedeckten Karten in jeder Zeile und jeder Spalte gerade. Wird eine Karte (hier das Pik As) umgedreht, so wird die Anzahl der aufgedeckten Karten in genau einer Zeile (hier der zweiten) und einer Spalte (hier der vierten) ungerade.

meisten Leserinnen und Lesern ist dieses Beispiel sicher aus *Computer Science Unplugged* bekannt.[3]

In einer weiteren Kompetenzstufe steht die Verschlüsselung von Daten im Fokus:

> **MI2.1.2c** Die Schülerinnen und Schüler können Daten mittels selbstentwickelter Geheimschriften verschlüsseln.

Zentral ist hier das Adjektiv »selbstentwickelt« – ein ausdrücklicher Auftrag, kreatives Arbeiten ins Zentrum zu stellen. Die intrinsische Motivation der Schülerinnen und Schüler ist kaum irgendwo so groß wie bei diesem Thema. Aufbauend auf Ideen, die die Menschheit seit vielen Jahrhunderten begleiten (beispielsweise der *Cäsar*-Verschlüsselung, welche auch in Sprach- und Mathematiklehrmitteln eine feste Stelle besitzt, *Skytale* oder der *Freimaurerverschlüsselung*) werden Geheimschriften kreiert, die dann von Mitschülerinnen und -schülern geknackt und verbessert werden können.

[3] https://www.csunplugged.org/en/at-a-distance/parity-magic/

Algorithmen und Algorithmisches Denken in der Volksschule

Unter dem Begriff *algorithmisches Denken* werden eine Reihe von Problemlösekompetenzen wie *Abstraktionsvermögen*, *Zerlegung eines Problems in Teilprobleme* oder auch *(clevere) Informationsdarstellung* zusammengefasst; letztere bildet einen zentralen Teil des Prozesses in Abbildung 1.

Viele dieser Problemlösekompetenzen sind nicht einzig und allein in der Informatik zu finden – gut so, wir möchten ja auch über fächerübergreifenden Unterricht sprechen. Das Ergebnis der Anwendung dieser Problemlösekompetenzen ist in der Regel ein Algorithmus, der beliebige Instanzen eines gegebenen Problems lösen kann. Ein solcher Algorithmus besteht aus einfachen, kleinen Schritten, deren Ausführung keinen Intellekt benötigt. Auf der anderen Seite ist das Formulieren eines Algorithmus unter Verwendung der oben erwähnten Problemlösefähigkeiten kreative und sinnstiftende Arbeit. Es geht entsprechend nicht darum, dass die Informatik in die Schule gekommen ist, um die Schülerinnen und Schüler »wie einen Computer denken zu lassen«, sondern ihnen näherzubringen, »wie eine Informatikerin bzw. ein Informatiker denkt«. Es geht um kreatives Problemlösen; es geht darum, die entstandene Lösung aber so gut verstanden zu haben und deswegen so sauber kommunizieren zu können, dass sie sogar einem Computer in einer Programmiersprache »erklärt« werden kann. Natürlich denkt man nun sofort an *Generative AI* und *Prompt Engineering* (welches ebenfalls Aspekte des algorithmischen Denkens benötigt). Dennoch vertritt der Autor die Position, dass Grundlagen des Programmierens erlernt werden sollten, bevor im Unterricht Sprachmodelle zur Codeerzeugung eingesetzt werden.

Heute, da Computer etwa durch Sprachassistenten, Chatbots und eben die jüngsten großen Sprünge in *Generative AI* immer mehr vermenschlicht werden, ist es umso wichtiger, sich den Unterschied zwischen Computern und Menschen bewusst zu machen. Dies findet sich im Lehrplan 21 als Kompetenzstufe deutlich gefordert:

> **MI2.2.2e** Die Schülerinnen und Schüler verstehen, dass ein Computer nur vordefinierte Anweisungen ausführen kann und dass ein Programm eine Abfolge von solchen Anweisungen ist.

Um sich dies zu vergegenwärtigen, können die Schülerinnen und Schüler durchaus einmal die Rolle des Computers einnehmen und bewusst jede Improvisation bzw. das »Lesen zwischen den Zeilen« ausdrücklich unterbinden; beispielsweise im Sprachunterricht:[4]

1. Eine Schülerin und ein Schüler, die sich zunächst in zwei unterschiedlichen Räumen befinden, bilden ein Zweierteam. Der Schülerin wird nun eine Figur aus Bausteinen präsentiert, etwa ein kleiner Turm aus fünf oder sechs Steinen. Die Aufgabe der

[4] https://algdenken.phgr.ch

Schülerin ist es, dieses Bauwerk so zu beschreiben, dass es aus denselben Steinen nachgebaut werden kann – Zeichnungen sind nicht erlaubt.
2. Nach zehn Minuten sollte der Text fertig sein, das Bauwerk wird fotografiert und anschließend zerlegt.
3. Nun betritt der Schüler den Raum und hat weitere fünf Minuten Zeit, um den Turm einzig anhand der schriftlichen Beschreibung nachzubauen.

Was zunächst einfach scheint, entpuppt sich dann oftmals als doch nicht ganz so unkompliziert. »Setze den kleinen blauen Stein auf den großen roten Stein« enthält doch noch einiges an Interpretationsspielraum und am Ende sieht der Turm ganz anders aus als auf dem Foto.

Das Fazit ist, dass es gar nicht so simpel ist, sich eindeutig auszudrücken, insbesondere wenn der Kommunikationspartner nichts anderes tut, als Eins zu Eins die aufgeschriebenen Schritte zu befolgen. Und auch wenn Computer, wie bereits erwähnt, stetig besser darin werden, uns vorzumachen, dass wir mit ihnen wie mit Menschen kommunizieren können, so braucht ihre Programmierung dennoch eine eindeutige Sprache.

Programmieren in der Volksschule

Apropos: Programmieren ist wahrscheinlich der prominenteste Aspekt der Informatik – und vielleicht auch (fälschlicherweise) einer der abschreckendsten. Dabei eignet sich gerade dieses Thema für ein Spiralcurriculum, das früh beginnt, spielerisch wichtige Konzepte vorstellt und nach und nach vertieft.

Wichtig ist in diesem Kontext der Begriff der *konzeptuellen Maschine* (»Notional Machine« (du Boulay et al., 1981)), welche sich aus den bekannten Befehlen und Konstrukten, die die verwendete Programmiersprache zur Verfügung stellt, ergibt. Diese abstrakte Maschine unterscheidet sich in der Regel stark von der *tatsächlichen* bzw. *ausführenden Maschine*, also dem Computer, den wir programmieren. Es ist offensichtlich keine gute Idee, den Schülerinnen und Schülern der Volksschule als ersten Kontakt mit der Programmierung den Befehlssatz aktueller Prozessoren vorzustellen. Besser beginnen wir mit einer einfachen konzeptuellen Maschine, welche die Programmiererin und der Programmierer bei ihrer Arbeit beobachten können. In der Programmiersprache *Logo* wird hierzu eine Schildkröte (die »Turtle«) verwendet, in *Scratch* z. B. eine Katze, in *Greenfoot* z. B. Wombats. Diese bieten den Programmiererinnen und Programmierern jeweils eine Identifikationsfigur: Die Kinder programmieren nicht mehr den Computer, sondern die »Turtle«, »Katze« oder den »Wombat«.

Die Logo-Turtle verfügt beispielsweise über einen sehr überschaubaren Satz von Befehlen, mit denen sie vorwärts und rückwärts laufen und sich nach links und rechts drehen kann; dabei zeichnet sie den zurückgelegten Weg. Wenige und einfache Befehle bedeuten, dass wir ein komplexeres Problem in kleinen, verständlichen Schritten lösen (müssen) und nicht durch Druck auf einen vorgefertigten Knopf, hinter dem alle wesentlichen Ideen verborgen bleiben. (Selbstverständlich werden auch hier aus einer simplen Vorwärtsbewegung unzählige Befehle in Maschinencode.)

Abbildung 3: Blockbasierte Zeichnung eines Vierecks in XLogoOnline Mini.

Jetzt ist wieder die bereits erwähnte präzise Kommunikation erforderlich; es wird genau das ausgeführt, was programmiert wird. Aber im Gegensatz zum Prozessor des Computers kann die Turtle nun dabei beobachtet werden; statt einer *Blackbox* ist sie eine *Glasbox*. Zu jedem Zeitpunkt sehen wir ihren Zustand: die Position, an der sie steht, in welche Richtung sie blickt, mit welcher Farbe sie zeichnet etc.

Für das Programmieren spricht der Lehrplan 21 über konkrete Konzepte, die die Schülerinnen und Schüler in Code umsetzen können sollen:

> **M2.2.2f** Die Schülerinnen und Schüler können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen.

Über die zu verwendende Programmiersprache werden hingegen keine Vorgaben gemacht und natürlich gibt es auch keine »beste« Programmiersprache. Wenn bereits in den ersten Schulstufen einfache Programme entwickelt werden sollten, bieten sich blockbasierte Programmiersprachen an; Abbildung 3 zeigt beispielsweise ein Programm, mit welchem die oben erwähnte Logo-Turtle in *XLogoOnline*[5] *Mini* ein einfaches Viereck zeichnet. Wenn die Schülerinnen und Schüler die Tastatur beherrschen, ist es möglich, zu einer textbasierten Sprache zu wechseln. Zum idealen Zeitpunkt, zu dem dieser Schritt genau erfolgen sollte, gibt es unterschiedliche Meinungen; der vorgestellte Ansatz geht von einem Wechsel in der fünften oder sechsten Klasse aus.

Auf eine Besonderheit sei aufmerksam gemacht: Um in gängigen imperativen textbasierten Programmiersprachen das Konzept der Schleife zu verwenden, muss in der Regel das Konzept der Variablen eingeführt worden sein. Dieses ist allerdings alles andere als einfach und die Wurzel vieler Fehlvorstellungen (Kohn, 2017). Im Lehrplan 21 werden Schleifen vor Variablen erwähnt; und von »allgemeinen« Variablen wird erst gesprochen, nachdem Parameter als eine Art eingeschränkte Variablen thematisiert wurden. Es ist deswegen sinnvoll, es den Schülerinnen und Schülern zu erlauben, Schleifen ohne Variablen umzusetzen. In vielen blockbasierten Sprachen, aber auch in textbasiertem Logo (z. B. *XLogoOnline maxi*) ist dies mit der sogenannten `repeat`-Schleife möglich. Mit *TigerJython* bzw. *WebTigerPython*[6] existiert außerdem ein

[5] https://xlogo.inf.ethz.ch

[6] https://webtigerpython.ethz.ch

Python-Dialekt, der diesen Schleifentyp übernommen hat und die Umsetzung eines Spiralcurriculums erlaubt:

- Erste Programmiererfahrungen werden bereits in den ersten zwei Klassen mit einer blockbasierten Logo-Variante gemacht, die im Wesentlichen keine Zahlen verwendet.
- Später wird das Prinzip des Parameters eingeführt, wenn die Turtle nicht mehr einfach nur »nach vorne« geht, sondern sich beispielsweise »100 Schritte (Pixel) nach vorne« bewegt.
- Es folgt ein Wechsel zu textbasiertem Logo, in dem die blockbasierten Konzepte nun wiederholt umgesetzt werden. Zunächst werden Befehle einzeln und dann Befehlssequenzen ausgeführt.
- Als nächstes Konzept wird die *Modularisierung* in Form von Unterprogrammen eingeführt. Hier werden zum ersten Mal Parameter definiert und nicht bloß in vorgefertigten Befehlen verwendet.
- Später (etwa ab der Sekundarstufe I) tritt Python (z. B. *WebTigerPython*) an die Stelle von Logo.
- Zu Beginn der Sekundarstufe II hatten die Schülerinnen und Schüler somit bereits Kontakt mit grundlegenden Programmierkonzepten und einer »richtigen« Programmiersprache, die heute in immer mehr modernen Anwendungen eingesetzt wird.

Informatik im Gymnasium

In Zukunft werden Lehrpersonen an Schweizer Gymnasien im Idealfall also eine Klasse voll mit Schülerinnen und Schülern vor sich haben, die bereits Grundlagen über die Verarbeitung von digitalen Daten besitzen. Sie haben beispielsweise bereits mit dem Binärsystem gearbeitet und kennen dessen Bedeutung für die Informatik; sie haben selbst Ideen zur Geheimhaltung von Texten entwickelt; sie haben Daten mit Prüfbits gegen Übertragungsfehler geschützt; und sie haben Grundkonzepte der Programmierung untersucht und umgesetzt.

Im Gymnasium können diese Konzepte nun vertieft werden, wobei stetig an das Vorwissen der Schülerinnen und Schüler angeknüpft wird und weitere Brücken zu Themen innerhalb und außerhalb der Informatik gebaut werden. Parallel zum Aufbau der Kompetenzen in Mathematik können Grundkonzepte der Informatik weiter formalisiert werden. Es folgen zwei Beispiele, die einen Bogen zur Volksschule spannen.

Darstellung von Daten im Gymnasium

Auch für ältere Schülerinnen und Schüler haben Geheimschriften keineswegs ihren Reiz verloren. Hier kann eine Brücke zur Programmierung geschlagen werden, indem beispielsweise die bereits bekannte *Cäsar*-Verschlüsselung programmiert wird. Nach der Vorstellung des ASCII- bzw. Unicodes kann ein gegebener Text zum Beispiel mit wenigen Zeilen Python-Code einfach entschlüsselt werden; siehe Abbildung 4.

```
text = "TYQZCXLETVTDEVCPLETGPLCMPTE"

for k in range(0, 25):
    for i in range(0, len(text)):
        print(chr((ord(text[i]) - 65 - k) % 26 + 65), end="")
    print()
```

Abbildung 4: Entschlüsselung eines mit Cäsar verschlüsselten Texts durch das Ausprobieren aller 25 möglichen Schlüssel. Angenommen wird hier, dass der Text aus den 26 Großbuchstaben des lateinischen Alphabets besteht.

Das Entschlüsseln von Geheimschriften kann genauer unter die Lupe genommen werden und über das bloße Ausprobieren (wie in Abbildung 4) hinausgehen. Hierzu können einerseits statistische Eigenschaften typischer Text ausgenutzt werden und andererseits kann nach Möglichkeiten gesucht werden, dies zu verhindern.

In der Kryptographie kann nun außerdem der Schritt hin zu moderneren Systemen gemacht werden – etwa mit *Public-Key-Kryptographie* und deren teilweise überraschenden Ideen. Realistischerweise kann hier zwar nicht auf die letzten Details der zugrundeliegenden Zahlentheorie eingegangen werden, dafür kann aber ein zweifelsfreier Wirklichkeitsbezug abgeleitet und nachvollzogen werden. In den Lehrplänen für das *Grundlagenfach Informatik*, das in der Schweiz ab August 2024 in die Schulen kommt, findet der Oberbegriff *Cyber Security* einen festen Platz, und somit können viele Konzepte im Unterricht besprochen werden, die auf asymmetrischer Verschlüsselung basieren.

Ähnliches gilt für die anderen eingangs erwähnten »Probleme« bei der Darstellung (und Verarbeitung) von Daten, nämlich die *Resistenz gegen Fehler* und *Effizienz der Darstellung*. Dort wiederum werden die in der Volksschule betrachteten Konzepte nun zu *Hamming-Distanz* und *Huffman-Code*. Die Aufgaben, Daten sicher vor unautorisiertem Zugriff zu halten, sie vor Übertragungsfehlern zu schützen und sie speichersparend zu kodieren, sollten möglichst »hands-on« angegangen werden.

Welche Funktionalität stellt ein Betriebssystem zur Verfügung? Wie kommunizieren Rechner miteinander? Hierbei geht es um nicht weniger, als darum, viele uns umgebende Alltagsphänomene besser zu verstehen (und kritisch zu hinterfragen!), Neugierde zu wecken und möglichst vielen Schülerinnen und Schülern den Wunsch nach und die Fähigkeit zur Mitgestaltung mitzugeben.

Programmierung und Algorithmik im Gymnasium

An dieser Stelle kann damit begonnen werden, die Programmierung und die Algorithmik zusammenzuführen, also die erstellten Algorithmen formaler zu untersuchen. Hier findet zum ersten Mal der Begriff der *Zeitkomplexität* einen Platz. Dieser wird eingeführt, ohne zu genau auf das konkrete Maschinenmodell einzugehen – es wird weiterhin eine *konzeptuelle Maschine*

```python
def binsearch(data, searched):
    left = 0
    right = len(data) - 1
    while left <= right:
        current = (left + right) // 2
        if data[current] == searched:
            return current
        elif data[current] > searched:
            right = current - 1
        else:
            left = current + 1
    return -1

print(binsearch([2,3,5,7,11,13,17], 3))
```

Abbildung 5: Die binäre Suche in Python.

verwendet, bei der sich beispielsweise noch keine großen Gedanken darüber gemacht wird, wie viel länger eine Multiplikation als eine Addition dauert. Bereits für die Volksschule findet sich im Lehrplan 21 hierzu (wohlgemerkt gegen Ende und hinter dem sogenannten *Grundanspruch*) folgende Kompetenzstufe:

> **MI.2.2.3i** Die Schülerinnen und Schüler können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z. B. lineare und binäre Suche, Sortierverfahren).

Eine Möglichkeit, im Gymnasium Algorithmen in einer »echten« Programmiersprache umzusetzen, Komplexitätsbetrachtungen zu erlauben und gleichzeitig an das vorhandene Wissen anzuknüpfen, ist durch den Python-Dialekt, der bereits im vorangegangenen Abschnitt besprochen wurde, gegeben. Dieser beinhaltet ebenfalls »die Turtle«, die dann nach und nach durch abstrakte Daten und Objekte ersetzt werden kann. Sind Parameter, Variablen, bedingte Anweisungen und Schleifen jetzt bekannt, können nun einfache Datenstrukturen eingeführt werden, beispielsweise Listen.

Die oben erwähnte *binäre Suche* ist eines der bekanntesten Beispiele für eine clevere Lösungsstrategie, welche die Lernenden idealerweise bereits in den ersten Jahren ihrer Schullaufbahn kennengelernt haben – ganz ohne Computer. Eine beispielhafte Implementierung in Python ist in Abbildung 5 gezeigt.

Der Umsetzung ging ein Herleiten (Wiederholen) in Alltagssprache, die Stück für Stück weiter formalisiert wurde, voraus. Das Entwickeln des Codes wird außerdem von intensiven Tests begleitet; gerade die binäre Suche ist zwar recht schnell erklärt, aber viel weniger schnell korrekt umgesetzt (Bentley, 1999). Auf dieser Schulstufe wird die Korrektheit des Algorithmus genauer untersucht, indem die Schülerinnen und Schüler beispielsweise begründen, dass der Algorithmus sicher *terminiert*, also nicht für immer läuft. Auch die Zeitkomplexität kann betrachtet werden, wobei hier der wesentliche Punkt ist, dass der *Suchraum*, also der Bereich, in dem sich die gesuchte Zahl noch befinden könnte, mit jeder Ausführung der Schleife ungefähr halbiert wird – falls das Element nicht bereits gefunden wurde. Auch ganz ohne *Landau-Symbole* und *Rekursionsgleichungen* kann so eine gute Intuition für die logarithmische Zeitkomplexität des Algorithmus hergeleitet werden; im Kern steht die Frage: »Wie oft muss ich die gegebene Liste halbieren, bis nur noch ein Element übrig ist?«

Ähnliche Unterrichtssequenzen können einfache Sortieralgorithmen behandeln oder sich mit der Suche nach Mustern in Daten befassen. Auch hier ist es möglich (und wünschenswert), die entwickelten Algorithmen hinsichtlich Korrektheit und Laufzeit zu untersuchen.

Informatik nach dem Gymnasium

Was wird Dozierende an Schweizer Hochschulen also im Idealfall in ein paar Jahren erwarten? Erstsemesterstudierende, für die die binäre Suche ebenso zum Allgemeinwissen gehört wie Schillers Tell, die Photosynthese oder die erste Ableitung von $f(x) = 2x + 5$. Allgemein sind Techniken trainiert worden, die es erlauben, Problemlösestrategien anzuwenden und das Ergebnis formal und präzise zu kommunizieren. So kann in den Informatik-Grundlagenvorlesungen, die womöglich bald in vielen Studienrichtungen einen ähnlichen Stellenwert wie Mathematik-Grundlagenvorlesungen besitzen werden, bereits ein hohes Niveau erreicht werden. Der Fokus an den Hochschulen kann zudem auf den konkreteren Einsatz von Informatikkonzepten in den jeweiligen Studienrichtungen gerichtet werden – etwa, wenn in der eingangs erwähnten Vorlesung zur Humanmedizin untersucht wird, wie gewisse Voraussagen über bestimmte Gewebeproben gemacht werden können. Eines der Hauptziele muss es sein, dass Hochschulen Studierende in den Hörsälen sitzen haben, die ein korrektes Bild der Informatik haben, und die sowohl ihren Nutzen als auch die nicht zu bestreitende Eleganz ihrer Konzepte kennen.

Jedoch möchte ich mit einer anderen Gruppe von Personen schließen, die – zwar immer wieder erwähnt – in diesem Artikel bislang zu kurz kam; und das, obwohl sie einen der kritischsten Teile unserer Gesellschaft ausmacht: die Lehrpersonen, sowohl in der Volksschule und dem Gymnasium als auch an den Hochschulen. Wenn wir von »Informatik nach dem Gymnasium« sprechen und hier das Bild einer nicht zu fernen Zukunft zeichnen, so muss natürlich gesagt werden, dass die gesamte Vision mit den Lehrpersonen steht und fällt. Und nun haben die dienstälteren unter ihnen wiederum womöglich weder vor, noch während, noch nach dem Gymnasium besonders viel Kontakt mit der Informatik gehabt. Hier sind die Institutionen *nach dem Gymnasium* in der Pflicht.

Gerade Lehrpersonen der Volksschule sind (in der Schweiz) Generalistinnen und Generalisten. Sie hatten wahrscheinlich keine Informatik in der Schule, womöglich auch nicht im Studium und brauchen deswegen für den Unterricht eines Moduls bzw. Fachs, das es zu ihrer eigenen Schulzeit nicht gegeben hat, die Unterstützung *nach dem Gymnasium*. Hier sind gerade die pädagogischen Hochschulen gefragt; es muss beispielsweise sichergestellt werden, dass, nachdem die obligatorischen Lehrplan-21-Weiterbildungen abgeschlossen sind, freiwillige Weiterbildungsprogramme angeboten werden, die interessierten Lehrpersonen (und davon wird es immer genug geben) die Werkzeuge an die Hand geben, um einen sinnstiftenden und nachhaltigen Informatikunterricht durchzuführen. Hier gibt es mittlerweile einiges an Bewegung mit Dozierenden, die wiederum eine formale Informatikausbildung hinter sich haben.

Für Gymnasiallehrpersonen ist die Situation anders, aber dennoch an manchen Stellen vergleichbar. Die typische Schweizer Informatiklehrerin und der typische Schweizer Informatiklehrer haben einen Informatik-Masterstudiengang absolviert und anschließend ein Lehrdiplom erhalten. Aber nun werden schnell Lehrpersonen benötigt, weswegen das *GymInf*-Programm[7] (kurz für »Informatikausbildung für Gymnasiallehrkräfte«) ins Leben gerufen wurde. Auch hier haben wir es mit einem ambitionierten Unterfangen zu tun, wenn im Rahmen von 107 ECTS-Punkten zum Beispiel aus der Biologie- auch eine Informatiklehrperson gemacht werden soll. Nach einigen Semestern und vielen Interaktionen mit den äußerst motivierten Teilnehmenden kann aber auch hier ergänzt werden: ambitioniert – aber keineswegs unrealistisch.

Literaturverzeichnis

Bentley, J. (1999): *Programming Pearls*, zweite Auflage, Addison-Wesley Professional.

du Boulay, B., O'Shea, T. & Monk, J. (1981): The black box inside the glass box: presenting computing concepts to novices, *International Journal of Man-Machine Studies* 14(3), pp. 237–249.

Hauser, U., Hromkovič, J., Klingenstein, P., Lacher, R., Lütscher, P. & Staub, J. (2020): *Einfach Informatik – Rätsel und Spiele ohne Computer*, Klett und Balmer Verlag.

Kohn, T. (2017): Variable evaluation: an exploration of novice programmers' understanding and common misconceptions. In *Proc. of SIGCSE 2017*, pp. 345–350.

Lizenz

Kontakt

Dennis Komm

ETH Zürich

https://people.inf.ethz.ch/dkomm/, dennis.komm@ethz.ch

[7] https://www.unifr.ch/gyminf/de/