MIA21.CH
MEDIEN INFORMATIK ANWENDUNG

Nutze die Macht der Algorithmen

Zyklus 2

LP21: Informatik – Algorithmen
Version 09/2020

Impressum

Version

September 2020

Modulverantwortung

Meier Urs L., Pädagogische Hochschule Luzern

Hauswirth Michel, Pädagogische Hochschule Luzern

Review

Dr. Assaf Dorit, Pädagogische Hochschule Zürich

Im Zentrum von MIA21 steht die Zusammenarbeit und das gemeinsame Weiterentwickeln. Aus diesem Grund freuen wir uns über kritische Rückmeldungen und Hinweise auf Rechtschreibfehler genauso wie über freundliches Lob. Am besten funktioniert das über unser Rückmeldeformular:

https://tinyurl.com/mia21-rueckmeldung

Inhaltsverzeichnis

Modulziele

Nach der Bearbeitung des Moduls «Nutze die Macht der Algorithmen, Zyklus 2»

- kennen Sie das diesem Modul zu Grunde liegende Kompetenzprofil und den Bezug zum Lehrplan 21.
- kennen Sie relevante Begriffe der Algorithmik sowie Grundlagen der Programmierung und können diese anwenden.
- kennen Sie Methoden zur Umsetzung von informatischer Bildung mit und ohne digitales Gerät.
- können Sie einfache Flussdiagramme und Programme erstellen sowie diese mit den Schülerinnen und Schülern umsetzen.

Vorgehen

Lernphase	Inhalte	Nachweise
Lernphase A: Einführung	Kompetenzprofil Erste inhaltliche Übung	Zeitplan Standortbestimmung Notizen zur Übung
Lernphase B: Vertiefung	Fachwissenschaftlicher und fachdidaktischer Hintergrund Sichtung weiterführender Links und Literatur	
Lernphase C: Umsetzung	MIA21-Aufgabe bearbeiten Unterrichtsplanung	Aufgabeneinreichung: MIA21-Unterrichtsszenario
Lernphase D: Abschluss und Reflexion	Abschliessende Reflexion	Ergänzung zur Selbsteinschätzung

Lernphase A: Einführung

1 Darum geht's

- Sie kennen das Kompetenzprofil des Lehrplans 21 zu diesem Modul und haben darauf basierend Ihren persönlichen Lernstand eingeschätzt.
- Sie nutzen erste Aufgaben, um sich mit dem Thema und digitalen Geräten sowie Online- und Offline-Programmierumgebungen vertraut zu machen.
- Sie haben die Lerngruppe für einen Erfahrungsaustausch genutzt und sich darin auf die Form der Zusammenarbeit im MIA21-Modul geeinigt sowie einen Zeitplan festgelegt.

2 Einleitung ins Thema

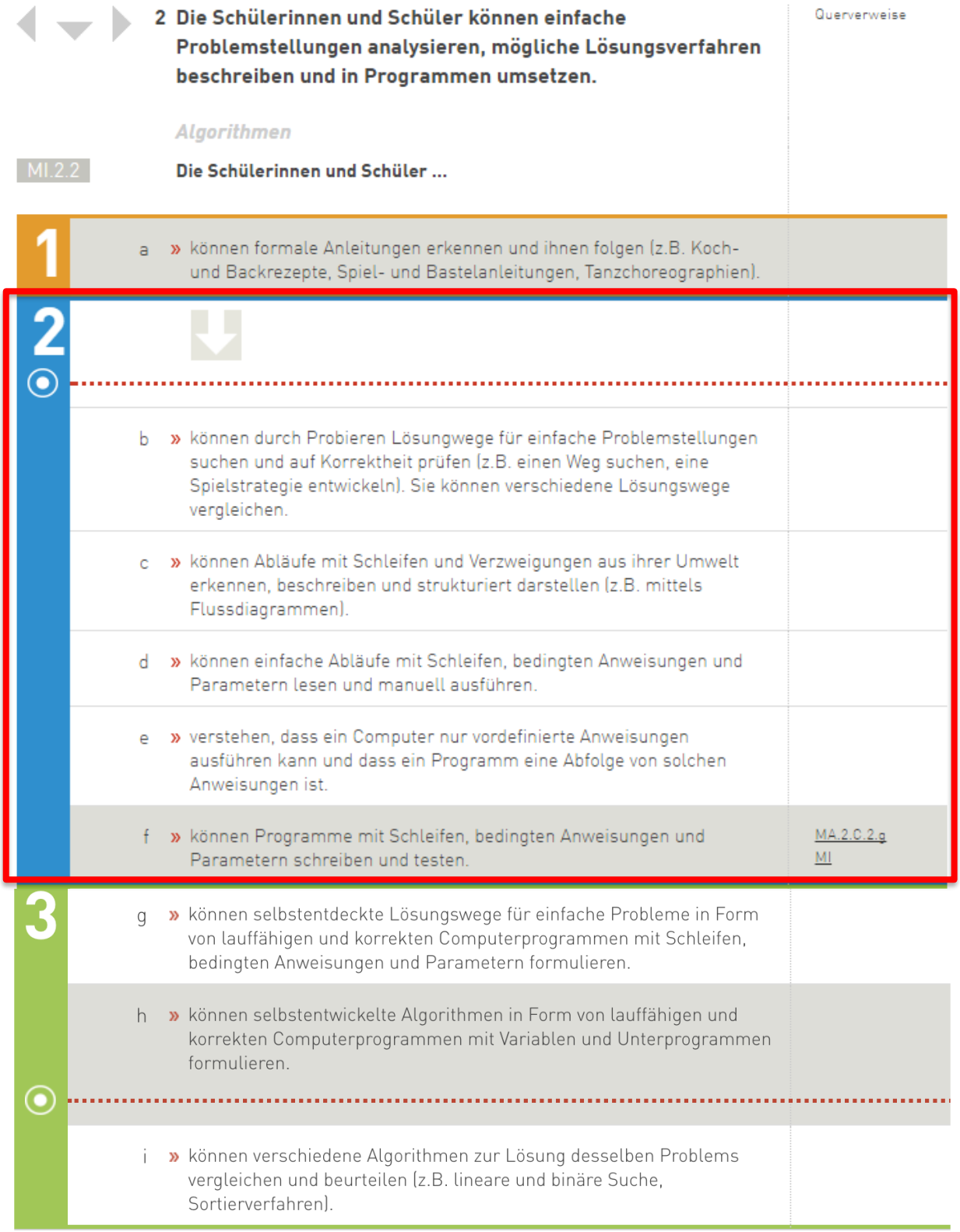
Im Sinne des Modultitels «Nutze die Macht der Algorithmen» ist es heute wichtig zu verstehen, wie die Programme funktionieren, um selber Einfluss nehmen zu können. Erst Programme machen digitale Geräte brauchbar und geben diesen ein scheinbar intelligentes Verhalten. Mit dem Verstehen von Algorithmen und der Erfahrung des Programmierens erhält man einen Einblick in die «Denkweise» digitaler Geräte. Im Modul werden die grundlegenden Funktionsweisen von Anweisungen in Programmen mit verschiedenen Programmiersprachen aufgezeigt.

3 Kompetenzen der Schülerinnen und Schüler gemäss Lehrplan 21

Die Auswahl der Themen in diesem Modul leiten sich aus dem Lehrplan 21 ab. Aufgaben und Beispiele wurden so gewählt, dass für die Schülerinnen und Schüler ein kontinuierlicher Aufbau von Informatikkompetenzen möglich wird.

Im Lehrplan 21 werden im Kompetenzbereich «Informatik» drei Teilkompetenzen unterschieden:

«Datenstrukturen» (Daten ordnen, strukturieren, darstellen), «Algorithmen» (Probleme schrittweise lösen) und «Informatiksysteme» (Computer und Netzwerke). Die einzelnen Teilkompetenzen sind nicht trennscharf, vielmehr braucht es für jede Tätigkeit in der Informatik alle drei Teile: die digitalen Geräte, die Algorithmen und die Daten, welche verarbeitet werden.



2 Die Schülerinnen und Schüler können einfache Problemstellungen analysieren, mögliche Lösungsverfahren beschreiben und in Programmen umsetzen.

Querverweise

Algorithmen

MI.2.2 **Die Schülerinnen und Schüler ...**

1

a » können formale Anleitungen erkennen und ihnen folgen (z.B. Koch- und Backrezepte, Spiel- und Bastelanleitungen, Tanzchoreographien).

2

b » können durch Probieren Lösungwege für einfache Problemstellungen suchen und auf Korrektheit prüfen (z.B. einen Weg suchen, eine Spielstrategie entwickeln). Sie können verschiedene Lösungswege vergleichen.

c » können Abläufe mit Schleifen und Verzweigungen aus ihrer Umwelt erkennen, beschreiben und strukturiert darstellen (z.B. mittels Flussdiagrammen).

d » können einfache Abläufe mit Schleifen, bedingten Anweisungen und Parametern lesen und manuell ausführen.

e » verstehen, dass ein Computer nur vordefinierte Anweisungen ausführen kann und dass ein Programm eine Abfolge von solchen Anweisungen ist.

f » können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen. — MA.2.C.2.g, MI

3

g » können selbstentdeckte Lösungswege für einfache Probleme in Form von lauffähigen und korrekten Computerprogrammen mit Schleifen, bedingten Anweisungen und Parametern formulieren.

h » können selbstentwickelte Algorithmen in Form von lauffähigen und korrekten Computerprogrammen mit Variablen und Unterprogrammen formulieren.

i » können verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen (z.B. lineare und binäre Suche, Sortierverfahren).

Abbildung 1 Lehrplan 21 «Medien und Informatik«, Teilkompetenz «Algorithmen, Zyklus 2».

4 Standortbestimmung

In der Standortbestimmung geht es um eine erste Kontaktnahme mit dem Programmieren. Im Rahmen dieser Aufgabe lernen Sie erste Algorithmen kennen:

- Gehen Sie auf die Seite von **Code.org**, klicken Sie auf «jetzt lernen» und wählen Sie unter «Stunde des Codes» eine der Varianten: Es spielt keine Rolle, für welche Sie sich entscheiden. Setzen Sie für das Lernprogramm ca. 30 Minuten ein.
- Diskutieren Sie in der Lerngruppe, welche Kompetenzen aus dem Lehrplan 21 im Bereich «Algorithmen» (siehe oben) mit dem Lernprogramm auf code.org gefördert werden.

Abbildung 2 Startseite Code.org (code.org, 2019).

Abbildung 3 Auswahl Lernprogramme Code.org (code.org, 2019).

5 Unterrichtsbezogene Annäherung ans Thema

Folgend finden Sie eine Lernaktivität für Schülerinnen und Schüler zum Thema «Algorithmus».

Probieren Sie die Aktivität aus und diskutieren Sie in der Lerngruppe Möglichkeiten, wie Sie diesen Auftrag oder ähnliche Aufträge in den Unterricht einbauen könnten.

Unsere Alltagswelt ist umgeben von Algorithmen. Es sind Bewegungs- bzw. Handlungsabläufe, die wir im Alltag ausführen und verinnerlicht haben. Sie sind im Gehirn abgespeichert. Für neue oder nicht gespeicherte Abläufe, wie spezielle Kochrezepte oder wie in unserem Beispiel ein Origami, benötigen wir eine Anweisung. Die Anweisung wird auch Bearbeitungsvorschrift oder Programm genannt. In einzelnen Befehlen, die in einer bestimmten Reihenfolge dargestellt werden, werden die Anweisungen abgearbeitet. Hier spricht man von linearer Anweisung (mehr dazu in den nächsten Kapiteln).

Die Abbildung 4 zeigt eine Anleitung für das Falten von **«Himmel und Hölle»**. Beschreiben Sie die Schritte 1–8 in Anweisungen. Versuchen Sie die Anweisungen so zu formulieren, dass ein digitales Gerät ihr folgen könnte. Finden Sie auch Wiederholungen von Anweisungen.

Beispiel Schritt 1:

- Das Blatt parallel zur einen Seite in der Mitte falten.
- Das Blatt 90 Grad im Uhrzeigersinn drehen.
- Das Blatt parallel zur einen Seite in der Mitte falten.

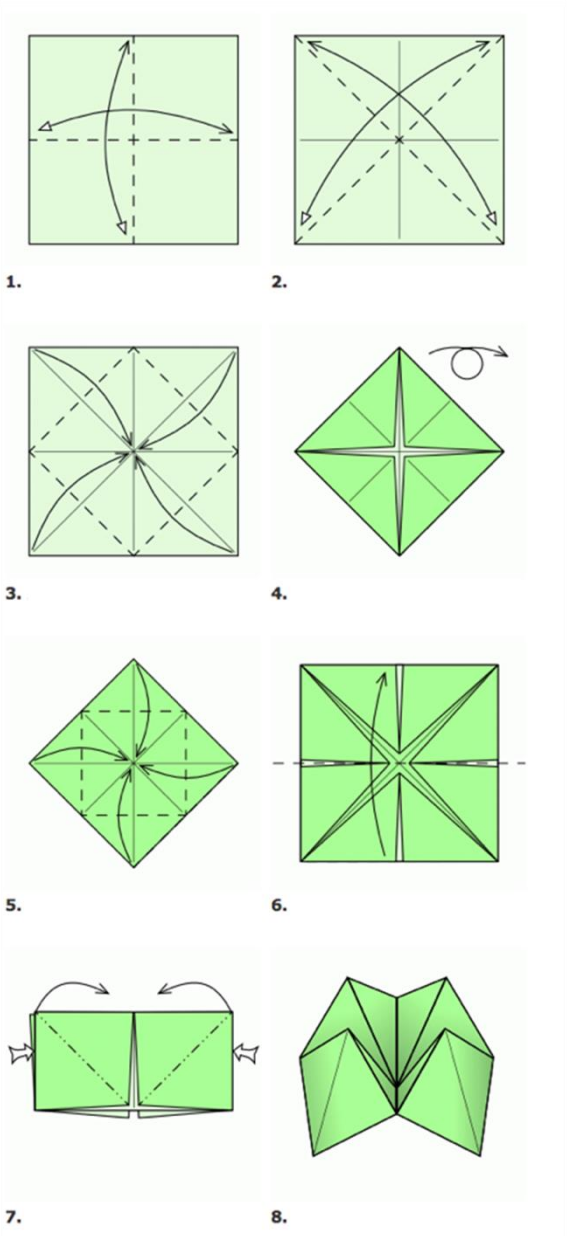


Abbildung 4 Anleitung «Himmel und Hölle» ohne Anweisungen (www.origami-kunst.de, 2017).

Lernphase B: Vertiefung

1 Darum geht's

- Sie sind vertraut mit den theoretischen Grundlagen zum Thema «Algorithmen, Zyklus 2».
- Sie haben erste Programmiererfahrungen in einer visuellen Programmierumgebung.
- Sie kennen didaktische Grundlagen des Programmierens und der Algorithmen.
- Sie kennen zur Thematik passende Lehrmittel/Websites wie: «Die Minibiber», «Informatik-Biber», «Primalogo» und «Computer-Science unplugged» sowie konkrete Unterrichtsideen dazu.

2 Fachwissenschaftlicher Hintergrund

2.1 Algorithmen

Weil man Algorithmen weder anfassen noch riechen kann, wissen vermutlich die wenigsten Menschen genau, was ein Algorithmus ist. Und dennoch sind sie allgegenwärtig: beim Wählen des Programms der Waschmaschine, beim Bezahlen der Parkgebühr oder beim Lösen eines Billetts für den Bus nach Hause.

Doch was genau ist ein Algorithmus?

Dies lässt sich am besten an einem einfachen Beispiel erläutern. Befolgen Sie dazu unten beschriebene Anleitung. Sie werden ein bisschen Kopfrechnen müssen. Aber keine Angst, es handelt sich um Grundoperationen im Zahlenraum bis 1000.

- Wählen Sie eine Zahl zwischen 1 und 9.
- Verdoppeln Sie die Zahl.
- Addieren Sie 2.
- Multiplizieren Sie die Zahl mit 100.
- Halbieren Sie das Resultat.
- Wenn Sie bereits Geburtstag hatten, addieren Sie das aktuelle Jahr und subtrahieren 2000.
- Wenn Sie noch nicht Geburtstag hatten, addieren Sie das aktuelle Jahr und subtrahieren 2001.
- Subtrahieren Sie die letzten beiden Zahlen Ihres Jahrganges (z.B. bei 1991 subtrahieren Sie 91).

Ihre Zahl sollte dreistellig sein. Die erste Ziffer besteht aus der Zahl, welche Sie sich am Anfang gemerkt haben, die letzten beiden Ziffern sind Ihr Alter in Jahren. Verblüffend, nicht? Doch darum geht es gar nicht. Wichtiger: Sie haben einen Algorithmus angewendet.

Gablers Wirtschaftslexikon definiert den Begriff Algorithmus wie folgt:

«Lösungsverfahren in Form einer Verfahrensanweisung, die in einer wohldefinierten Abfolge von Schritten zur Problemlösung führt» (Gabler Springler, 2016).

Sie haben sogar einen Algorithmus mit einer bedingten Anweisung («Wenn Sie bereits Geburtstag hatten, …») angewendet, weil Sie dort eine Entscheidung treffen mussten. Der obige Algorithmus verlangte nach einer Eingabe (Input) und verarbeitete diese Eingabe zu einer Ausgabe (Output). Die Verarbeitungsphase kann sowohl von einem Menschen als auch von einer Maschine ausgeführt werden, weil zu jedem Zeitpunkt klar ist, was zu machen ist.

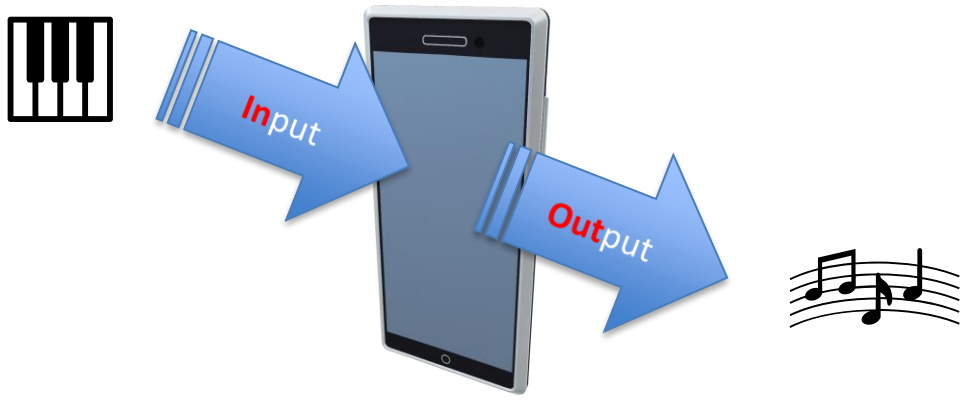


Abbildung 5 Eingabe – Verarbeitung - Ausgabe.

2.1.1 Einführungsbeispiel Algorithmus als «Scratch»-Programm, Struktogramm und Flussdiagramm

Abbildung 6 Beispiel Algorithmus als «Scratch»-Programm.

ALGORITHMUS

Zahl = 5
Geburtstag = True
Jahrgang =1974
aktuellesJahr = 2017
Zahl = 2 * Zahl
Zahl = 2 + Zahl
Zahl = 100 * Zahl
Zahl = Zahl / 2
Geburtstag == TRUE
V / F
Zahl = Zahl +aktuellesJahr – 2000 / Zahl = Zahl + aktuellesJahr – 2001
Zahl=Zahl – Jahrgang + 1900

Abbildung 7 Einführungsbeispiel als Struktogramm.

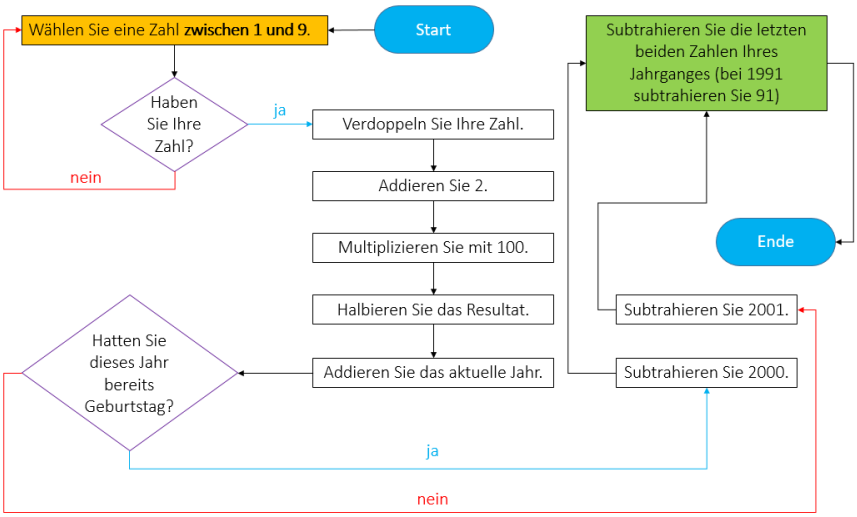


Abbildung 8 Einführungsbeispiel als Flussdiagramm.

Fehler! Verweisquelle konnte nicht gefunden werden. zeigt ein «Scratch»-Programm des Einführungsbeispiels zu Algorithmen. Zuerst wurden einige Variablen (Zahl, Geburtstag, Jahrgang und aktuelles Jahr) eingeführt und initialisiert, d.h. ihnen wurde bereits ein Wert zugewiesen. Das «wenn» stellt die bedingte Anweisung dar («Wenn Sie bereits Geburtstag hatten, …»). Hier wird nur die eine oder andere Berechnung durchgeführt. Die **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt das E inführungsbeispiel als Struktogramm und **Fehler! Verweisquelle konnte nicht gefunden werden.** als Flussdiagramm dargestellt. Nach dem Lesen des Kapitels 2.3 wird Ihnen der Aufbau der beiden Diagramme klar sein.

2.2 Datentypen

Der Algorithmus aus Kapitel 1 «Algorithmen», bei welchem Ihre Kopfrechenfertigkeiten geprüft wurden, berechnete eine dreistellige natürliche Zahl. Um diesen Wert abspeichern und zurückgeben zu können, benötigt man eine Variable. Jede Variable hat einen eigenen Datentyp.

Stellen Sie sich dazu ein «Mise en Place» beim Kochen vor. Hier legen Sie alles bereit, was Sie zur Zubereitung des Mittagessens benötigen. Neben Gemüse, Salz und Messern stellen Sie auch verschiedene Gefässe für Zutaten oder Ähnliches bereit. Die verschiedenen Gefässe stellen die verschiedenen Variablen dar, die Sie bei einem Programm benötigen. Jedes Gefäss hat genau deklarierte Inhalte. In das grüne kommt die geschnittene Zwiebel, im ovalen liegen Oliven bereit und im hohen Gefäss befinden sich frische Pilze. Der Verwendungszweck der Gefässe entspricht dem Datentyp. Jedes Gefäss ist für einen bestimmten Inhalt gedacht, genauso wie jede Variable etwas ganz Bestimmtes speichern soll. Also ist die Analogie des Inhaltes der Datentyp.

Der wichtigste Datentyp ist «Integer», er lässt das Speichern von ganzen Zahlen (also inklusive der negativen Zahlen) zu. Damit lassen sich ganze Zahlen zwischen -2^{31} und $2^{31} - 1$ (also zwischen rund -2 Milliarden und 2 Milliarden) speichern. Es gibt aber nicht nur numerische Datentypen für das Speichern von Zahlen mit Hilfe von Variablen, sondern auch alphanumerische, wie z.B. «Character» (char) oder «String», für das Speichern von einzelnen Buchstaben respektive von Zeichenketten variabler Länge. Die 9 zeigt Variablen (z.B. meinByte) mit entsprechendem Datentyp (z.B. byte) und einer Wertzuweisung (z.B. 01101011) als Schachteln dargestellt.

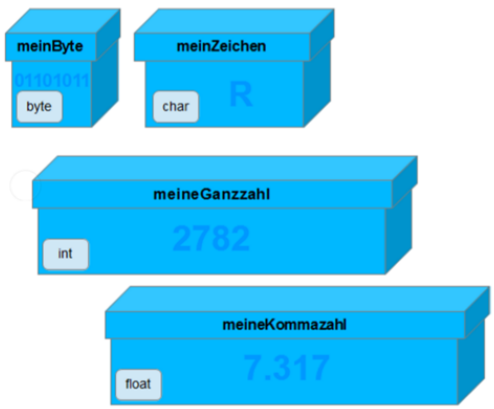


Abbildung 9 Verschiedene Variablen mit zugehörigem Datentyp als Schachtel dargestellt (media.kswillisau.ch, 2017).

.Tabelle 1: Datentypen und ihre Wertebereiche.

Datentyp	**Grösse in Bits**	**Wertebereich**
Wahrheitswert: *boolean*	1 bit	wahr/falsch (true/false)
Vorzeichenbehaftetes Byte	8 bit	0 bis 255
Kleine Ganzzahl: *short*	16 bit = 2 Byte	-32'768 bis 32'768
Ganzzahl: *integer*	32 bit = 4 Byte	ca. -2 Milliarden bis ca. 2 Milliarden
Zeichen: *character*	8 bit = 1 Byte	Zeichen des ASCII-Zeichensatz (siehe Modul I-2a: «Mit Daten jonglieren»)
Gleitkommazahl: *float*	32 bit	$-3 \cdot 10^{38}$ bis $3 \cdot 10^{38}$
Farbwert: *color*	3 · 8 bit = 3 Byte	2'777'216 Farben
Zeichenkette: String		Zeichenkette zwischen 0 und ca. 2 Milliarden Zeichen

Als Beispiel zeigt die nächste Abbildung eine textbasierte Programmiersprache. Es wird eine Funktion namens **erhoehen** dargestellt, welche zuerst die Variable **kleineZahl** mit dem Datentyp «short» einführt (deklariert) und dann die Zahl 3141 zuweist (initialisiert). Danach wird die Zahl um eins erhöht (inkrementiert; **kleineZahl++**). Mit **return** wird der unter den Variablen **kleineZahl** gespeicherte Wert zurückgegeben.

```
int erhoehen() {
   short kleineZahl;  // Variable kleineZahl mit den Datentyp short
   kleineZahl = 3141; // Variable kleineZahl wird mit dem Wert 3141 initialisiert
   kleineZahl++;      // Variable wird um eins inkrementiert (erhöht), hat also den Wert
   return kleineZahl; // Wert der Variable wird zurückgegeben
}
```

Abbildung 10 Funktion «erhoehen».

Höhere Programmiersprachen wie Java, C++, Python etc. verfügen über eine Vielzahl weiterer Datentypen.

2.3 Programmieren

Das Programmieren ist ein wichtiger Bestandteil für das Verständnis, wie digitale Geräte funktionieren. In erster Linie geht es beim Programmieren darum, dem digitalen Gerät zu sagen, was es tun soll. Die menschliche Sprache wird in einer formalen Sprache ausgedrückt, die dann durch die Kompilierung[1] in eine Maschinensprache (Binärcode) übersetzt wird.

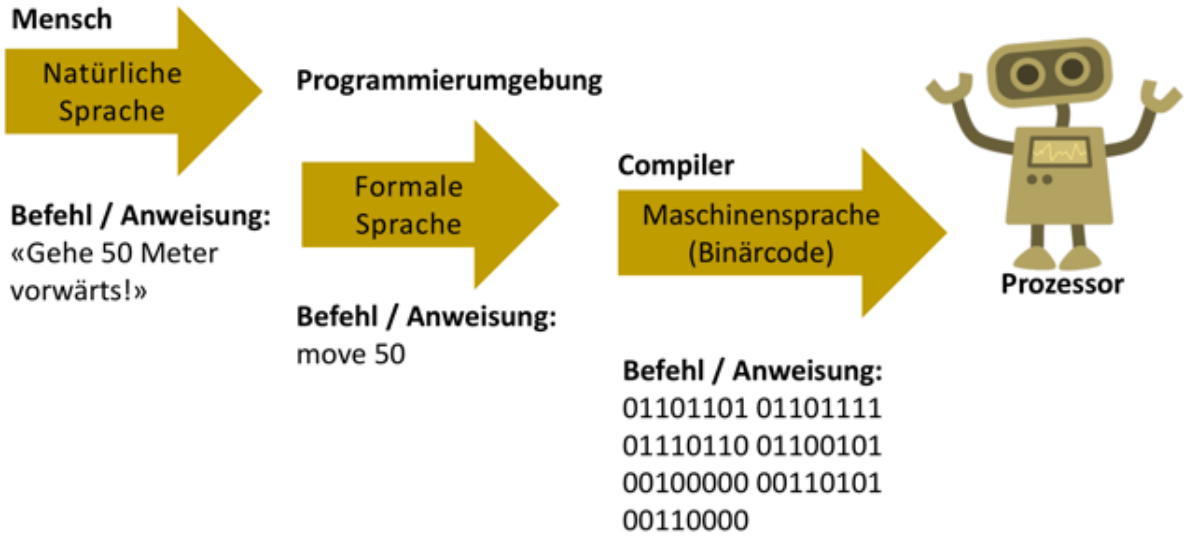


Abbildung 11 Vom Befehl zur Verarbeitung im Prozessor (PH Luzern, 2016).

Die Herausforderung besteht darin, die Befehle genau vom menschlichen Denken in eine formale Sprache zu übergeben. Darin spielt neben der genauen Formulierung auch die Logik eine wichtige Rolle. Wie im Kapitel «Algorithmus» erläutert, werden in Programmen genaue Rechenvorschriften festgelegt. Das Programm folgt genau diesen Rechenschritten. Der Prozess des Programmierens ist eine wichtige Erfahrung. Er stellt hohe Ansprüche in Bezug auf genaues Arbeiten, schrittweises Vorgehen, Aufteilung in kleinere Programme (Unterprogramme) sowie logisches Denken. Weiter wird die Ausdauer beim Lösen von Problemen geschult.

2.3.1 Programmieren – textbasiert und/oder visuell

Für den Zyklus 2 gibt es gute Programmierumgebungen, mit welchen textbasiert oder auch visuell programmiert werden kann. Der Einsatz hängt von der Stufe und den zu erreichenden Zielen ab.

Textbasierte Programmierumgebungen wie «Xlogo» oder «TigerJython» verlangen die genaue Eingabe mit der Tastatur und das Erlernen einer formalen Sprache. Sie eignen sich ab der 5. Primarklasse mit dem Vorteil, dass sie gut auf weiterführende Programmiersprachen vorbereiten.

Bei den visuellen Programmierumgebungen wie «Scratch», «TouchDevelop», «Lego» oder «ThymioVPL» können per Drag&Drop die Programmbausteine zusammengefügt werden. Hier liegt ein Schwer-

[1] Kompilierung wird der Vorgang genannt, der die formale Programmiersprache in eine Maschinensprache übersetzt.

punkt auf dem Programmablauf und dessen Logik. Die Syntaxfehler von textbasierten Programmierum-

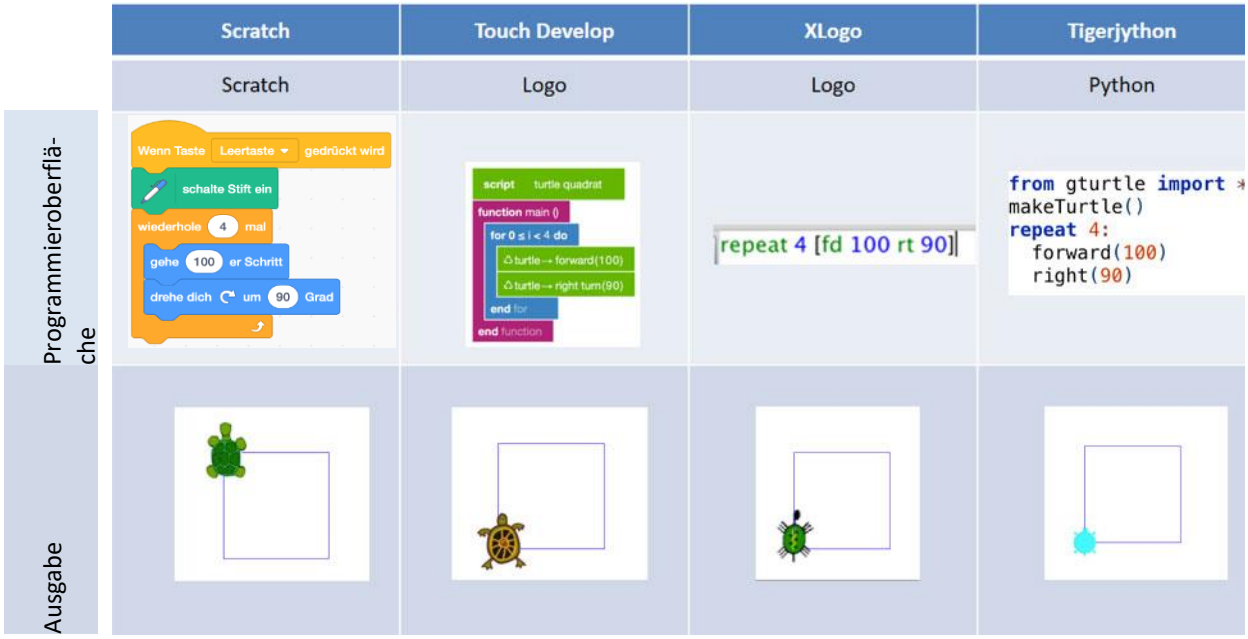


Abbildung 12 Übersicht didaktischer Programmierumgebungen (PH Luzern, 2016).

gebungen entfallen. Ein Einsatz von visuellen Programmierumgebungen ist ab der 3. Primarklasse sinnvoll.

2.3.2 Grundlagen des Programmierens

Die Programmanweisungen werden in verschiedene Kategorien aufgeteilt. Am besten wird dies mit einem Beispielprogramm von «Scratch» erklärt. Zu Beginn steht eine bedingte Anweisung. Die Bedingung prüft in diesem Falle, ob die Leertaste gedrückt ist. Anschliessend kommt ein erster Befehl. Weiter folgt eine Schleife, die vier Mal wiederholt wird. Innerhalb der Schleife gibt es wieder Befehle. Der ganze Ablauf wird als Algorithmus bezeichnet. Die **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt Programmierbausteine in visueller Sprache.

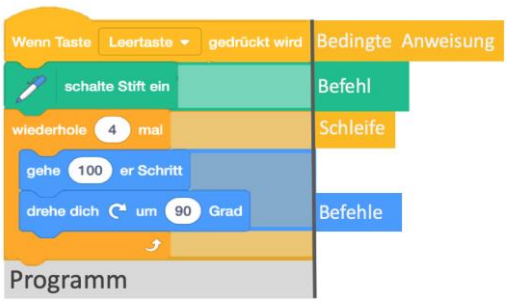


Abbildung 13 «Scratch»-Programm mit Programmierbausteinen.

Im nächsten Kapitel werden einzelne Programmierbausteine nochmals detailliert erklärt und darin ausschliesslich Begriffe besprochen, die eine hohe Relevanz für den Zyklus 2 haben und in den entsprechenden Lehrmitteln dieser Stufe eingeführt werden.

Hierfür werden die Programmieranweisungen jeweils zuerst in Worten, dann als Fluss- und Struktogramm sowie in der visuellen Programmierumgebung Scratch dargestellt. Die Reihenfolge von links nach rechts ist ein wichtiger Bestandteil der Programmierarbeit, damit werden Fehler in der Logik eines Programms reduziert. Bei den Diagrammen (Fluss- oder Struktogramm) **kann eines von beiden** eingesetzt werden. Das Flussdiagramm hat sich in der Praxis als einfacher erwiesen.

Anmerkung: Die unten aufgeführten Beispiele werden ohne Schleife/Wiederholung dargestellt und somit nicht wiederholt. Bei der Ausführung «Programm» in «Scratch» wird die Startbedingung 

«wenn angeklickt» weggelassen. Die Programmelemente in «Scratch» ganz rechts lassen sich in der Programmierumgebung durch Doppelklicken ausführen.

2.3.2.1 *Anweisungen – Sequenziell*

Die Befehle (Anweisungen A_1, A_2, A_3, …, A_n) werden in Folge einmal ausgeführt.

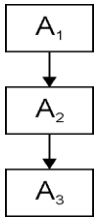
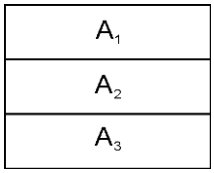
Anweisung in Worten	Flussdiagramm	Struktogramm	Programm in Scratch
A_1: 100 Schritte vorwärts A_2: Warte 1 Sek. A_3: 100 Schritte rückwärts			

Abbildung 14 Überblick Anweisung.

2.3.2.2 *Entscheidung – If/then/else*

Bei der Entscheidung findet immer zuerst eine Prüfung statt. Je nach Ergebnis wird dann «Wahr» (Ja) oder «Falsch» (Nein) ausgeführt.

Bei der Entscheidung gibt es zwei Varianten. Im ersteren Fall gibt es nur für das «Wenn wahr» eine Anweisung, im zweiten auch für das «Sonst».

Variante 1:

Einfache Auswahl («if/then»): Wenn die Bedingung B wahr ist, wird die Anweisung A_1 ausgeführt.

Anweisung in Worten	Flussdiagramm	Struktogramm	Programm in «Scratch»

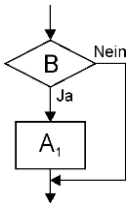
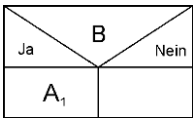
B: Pfeil gedrückt A_1: 100 Schritte vorwärts			

Abbildung 15 «if/then» – mit nur einfacher Anweisung.

Variante 2:

Zweifache Auswahl («if/then/else»): Wenn die Bedingung B wahr ist, wird die Anweisung A_1 ausgeführt; ist die Bedingung B falsch, wird die Anweisung A_2 ausgeführt.

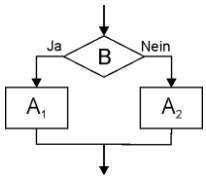
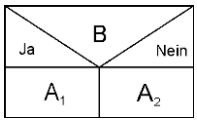
Anweisung in Worten	Flussdiagramm	Struktogramm	Programm in «Scratch»
B: Pfeil gedrückt A_1: 100 Schritte vorwärts A_2: 100 Schritte rückwärts			

Abbildung 16 «if/then/else»-Entscheidung.

2.3.2.3 Schleifen – Wiederholung

In Spielen werden weitgehend unendliche Wiederholungen benutzt. Oft werden aber Schleifen durch einen Event abgebrochen.

Flussgesteuerte Schleife («repeat-until»): Die Bedingung B wird geprüft. Erst dann wird die Bedingung A_1 ausgeführt. Wenn und solange B nicht wahr ist, wird Anweisung A_1 ausgeführt.

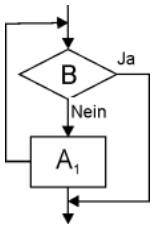
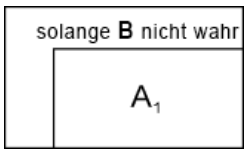
Anweisung in Worten	Flussdiagramm	Struktogramm	Programm in «Scratch»
A_1: 100 Schritte vorwärts B: Leertaste gedrückt?			

Abbildung 17 Schleife – Wiederholung.

2.3.2.4 *Unterprogramme*

Programme können auch in Unterprogramme aufgeteilt werden. Der Code wird dadurch übersichtlicher und die Unterprogramme können wiederholt und beliebig oft eingesetzt werden.

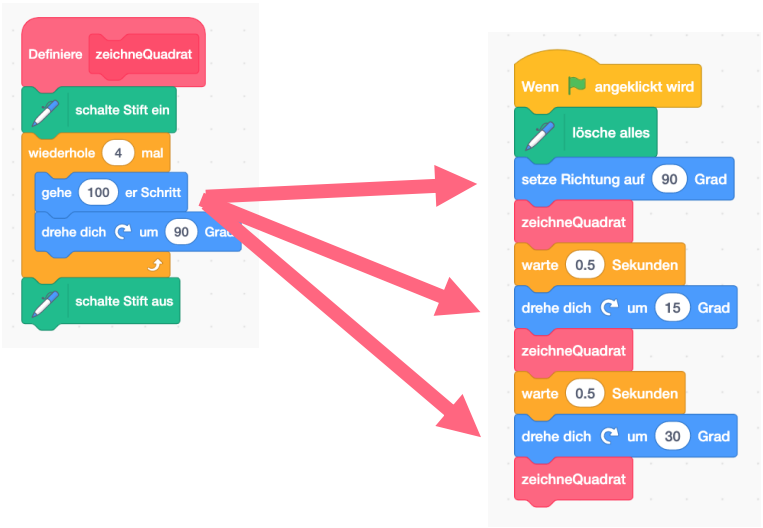


Abbildung 18 Unterprogramm mit Hauptprogramm.

2.3.2.5 *Parameter*

Spannend werden Unterprogramme, wenn sie individuell angepasst werden können. Dies lässt sich mit der Übergabe von Werten erreichen. Der Fachbergriff dazu heisst «Parameter».

Abbildung 19 Unterprogramm mit Parameter.

2.3.3 Programmieren lernen

Programmieren lernt man nur durch eigenes Tun. In diesem Sinne fordern wir Sie auf, selber zu programmieren. Die Aktivierung dient einerseits als fachliche Überhöhung, andererseits wird fachdidaktisch das Prinzip des selbständigen Entdeckens (siehe Grundlagenmodul) angewandt.

Gehen Sie zur Website scratch.mit.edu und klicken Sie auf «entwickeln».

Zeichnen Sie mit der Katze von «Scratch» ein Sechseck mit der Länge von 100 Pixel.

1. Verkürzen Sie den Code durch Wiederholungen/Schleifen.
2. Erstellen Sie für das Sechseck ein Unterprogramm, so dass Sie das Sechseck wiederholt einsetzen können.
3. Erstellen Sie ein Flussdiagramm für folgende Aufgabe:
   Beim Drücken der Taste «Q» wird ein Quadrat, ansonsten ein Dreieck der Seitenlänge 100 Pixel gezeichnet.
   Erstellen Sie dazu das «Scratch»-Programm.
4. Erweitern Sie das Programm aus Aufgabe 4 so, dass beim Drücken der Taste «L» ein Dreieck mit Seitenlänge 150 Pixel, ansonsten eines mit der Seitenlänge 100 Pixel entsteht. Arbeiten Sie nicht mit zwei Unterprogrammen, sondern mit Parametern.
5. Erweitern Sie das Programm aus Aufgabe 4 so, dass zusätzlich beim Drücken der Taste «K» ein Kreis mit dem Radius 100 Pixel gezeichnet wird.

Bemerkung:
Folgende Kompetenzen aus dem Lehrplan 21 werden mit diesen Aufgaben bereits aufgegriffen:
MI.2.2b, MI.2.2c, MI.2.2d, MI.2.2f (Grundanspruch Zyklus 2)

Die Lösungen zu diesen Aufgaben finden Sie auf dem YouTube-Kanal MIA21-Algorithmen im Zyklus 2 unter tinyurl.com/MIA21-I2b-1.

2.4 Algorithmen in der Praxis

2.4.1 Suchen

Ein Buch in der Bibliothek zu finden, kann ein mühsamer Prozess sein, obwohl die Bücher alphabetisch geordnet sind. Wie gehen Sie vor, wenn Sie ein bestimmtes Buch suchen? Laufen Sie so lange in der Bibliothek umher, bis Sie das Buch gefunden haben? Dann bevorzugen Sie die sequentielle Suche, wie die Abbildung 20 illustriert. Ein Buch wird mit dem nächsten verglichen, bis Sie am richtigen Ort landen.

Abbildung 20 Prinzip der sequentiellen Suche.

Eine andere Möglichkeit wäre, in der Mitte der Bücherreihe zu beginnen. Befindet sich das gesuchte Buch links vor diesem, gehen Sie in die linke Mitte, ansonsten in die rechte und so weiter. Diese Suche nennt man binäre Suche. Die Abbildung 21 zeigt die Idee dieser binären Suche. Doch welche Suche ist schneller respektive effizienter? Schnell heisst hier, dass möglichst wenige Vergleiche gemacht werden müssen. Der zurückgelegte Weg soll hier ausser Acht gelassen werden, obwohl dieser natürlich ebenfalls eine Rolle spielt.

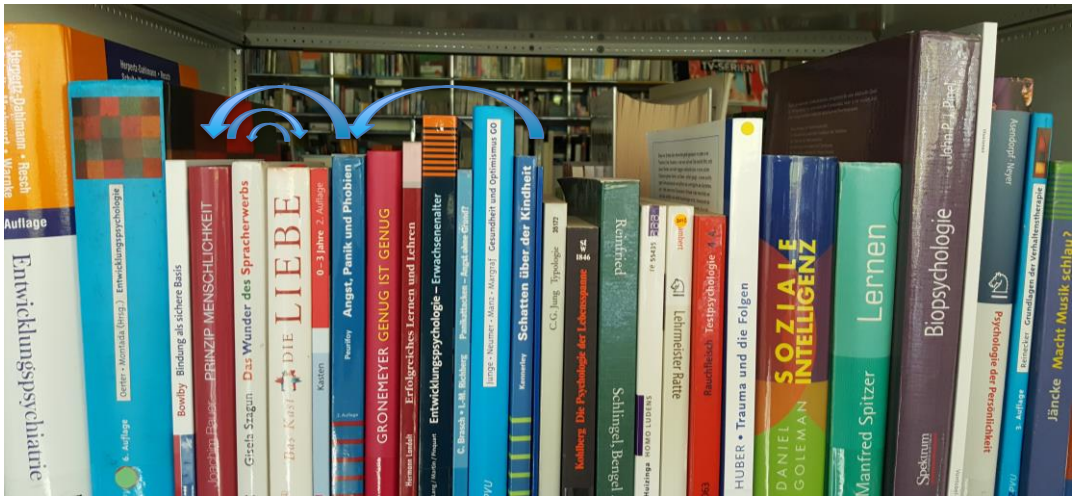


Abbildung 21 Prinzip der binären Suche.

Wenn es Sie interessiert, wie die binäre Suche mit einer höheren Programmiersprache umgesetzt werden könnte, sehen Sie folgend abgebildet eine Lösung in «Python».

```
def binsuch(werte, gesucht, start, ende) {
   if ende < start:
        return 'nicht gefunden'
   mitte = (start + ende)/2
   if werte[mitte] == gesucht:
        return mitte
   elif werte[mitte] < gesucht:
        return binsuch(werte, gesucht, mitte+1, ende)
   else:
        return binsuch(werte, gesucht, start, mitte-1)
```

Abbildung 22 Binäre Suche übernommen aus (wikipedia.org, 2017).

2.4.2 Sortieren

Sortieren heisst Ordnung halten und zwar so, dass die sortierten Objekte schnell wieder gefunden werden können. Denn sowohl Mensch wie auch Maschine tun sich mit sortierten Objekten leichter. Wie das Suchen funktionieren kann, haben Sie soeben gelesen. Zum Beispiel sind die Bücher in einer Bibliothek oder Namen von Lehrpersonen auf einer Telefonliste alphabetisch, die Briefe des Pöstlers nach Strassen und Hausnummern sortiert. So, wie es beim Suchen verschiedene Möglichkeiten gibt, existieren auch verschiedene Algorithmen zum Sortieren. Zwei mögliche Verfahren sollen hier konkret vorgestellt werden: die vergleichsbasierten Sortierverfahren «Insertion Sort» und «Bubble Sort».

2.4.2.1 Insertion Sort

«Insertion Sort» ist ein einfaches Sortierverfahren. In einer unsortierten Liste wird das erste noch nicht sortierte Element links ausgewählt und an der richtigen Stelle einsortiert, indem es jeweils mit den bereits sortierten Elementen verglichen wird. Ist das gewählte Element kleiner als der Vorgänger (Element links davon), werden die Plätze getauscht. So geht es nun weiter: Ist das gewählt Element kleiner als der Vorgänger (Element links davon), werden die Plätze getauscht und zwar so lange, bis das gewählte Element am richtigen Ort ist. Dann beginnt der Algorithmus von vorne.

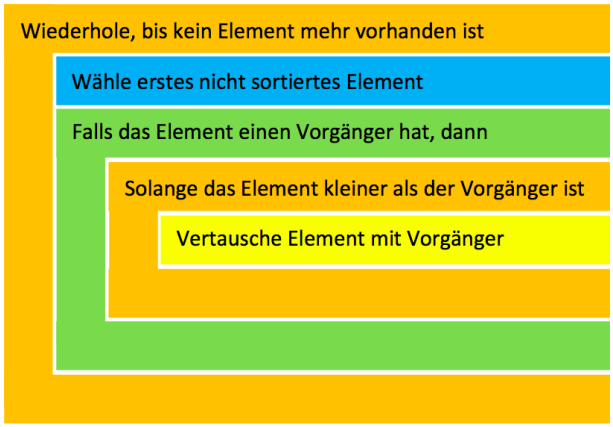
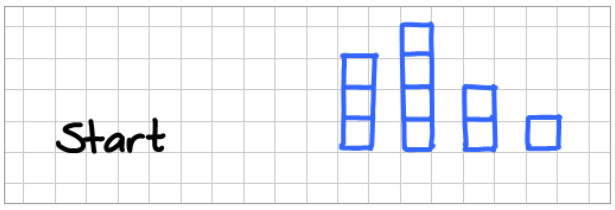
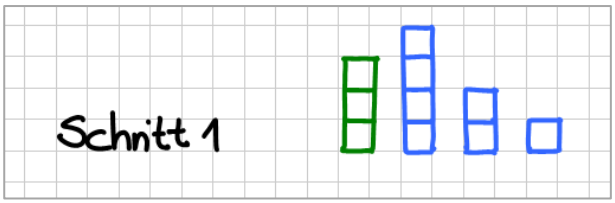


Abbildung 23 Schema «Insertion Sort».

Ein Beispiel soll den Algorithmus verdeutlichen. Wir starten mit vier Elementen, die der Grösse nach sortiert werden sollen. Das Element, welches gerade verarbeitet wird (d.h. ausgewählt wurde), ist grün dargestellt. Rot dargestellt ist jener Teil der Liste, welcher bereits sortiert ist.

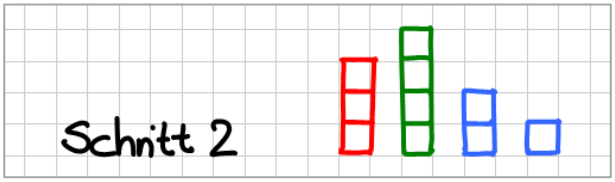


Schritt 1: Als Erstes wird das Element ganz links gewählt. Da das Element keinen Vorgänger hat, wird die Wiederholschleife erneut ausgeführt.



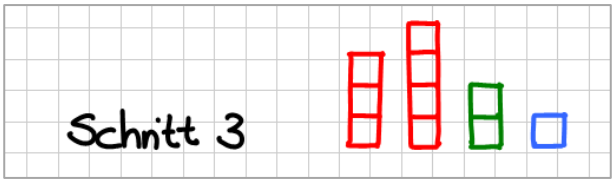
Anzahl Vertauschungen: 0

Schritt 2: Das zweite Element der Liste wird gewählt. Da der Vorgänger kleiner ist, ist das Element bereits am richtigen Ort. Es wird keine Vertauschung gemacht.



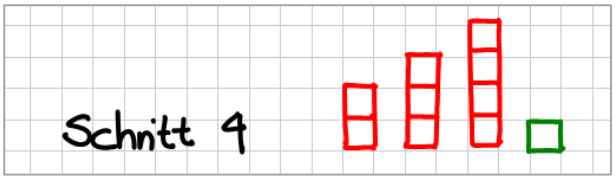
Anzahl Vertauschungen: 0

Schritt 3: Das dritte Element der Liste wird gewählt. Solange das Element kleiner als der Vorgänger ist, werden Vertauschungen gemacht. Das passiert zwei Mal.



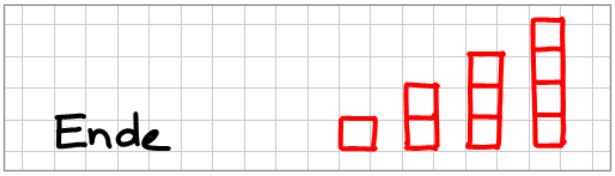
Anzahl Vertauschungen: 2

Schritt 4: Das letzte Element der Liste wird gewählt. Solange das Element kleiner als der Vorgänger ist, werden Vertauschungen gemacht. Das passiert drei Mal.



Anzahl Vertauschungen: 3

Nun ist die Liste vollständig sortiert.



In unserem einfachen Beispiel mussten bereits 5 Vertauschungen gemacht werden, um die Liste mit den vier Elementen zu sortieren.

2.4.2.2 *Bubble Sort*

Der «Bubble Sort»-Algorithmus durchläuft eine Liste jeweils von links nach rechts, vergleicht zwei benachbarte Elemente miteinander und vertauscht diese, falls nötig, wählt das nächste Element und vergleicht es mit dem Nachbarn, vertauscht, wenn nötig etc. Ist er am Ende der Liste angekommen, wiederholt er den Vorgang und beginnt wieder am Anfang der Liste, bis alle Elemente am richtigen Platz eingeordnet sind und keine Vertauschung mehr möglich ist. Der Algorithmus erinnert an das Aufsteigen von grossen Blasen, woher auch der Name «Bubble Sort» rührt. Im besten Fall (best case) wird die Liste nur einmal durchlaufen, nämlich dann, wenn die Elemente bereits sortiert sind. Im schlimmsten Fall (worst case) wird die Liste so oft durchlaufen, wie die Liste Elemente hat, und zwar dann, wenn die Liste verkehrt herum sortiert ist.

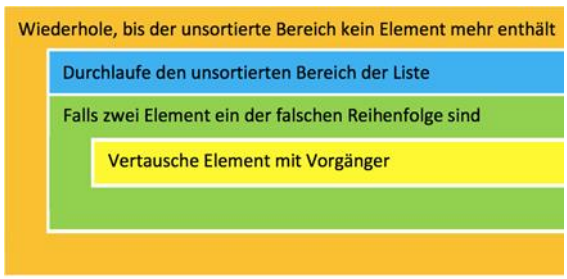
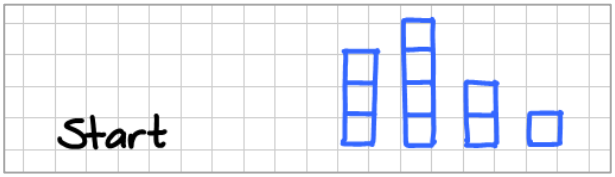
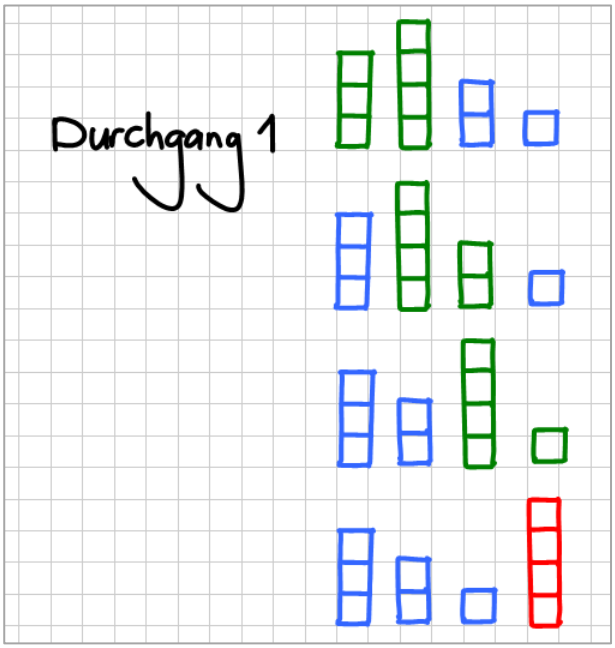


Abbildung 24 Schema «Bubble Sort».

Auch hier soll ein Beispiel den Algorithmus verdeutlichen. Wir starten mit derselben Liste wie beim «Insertion Sort». Der unsortierte Bereich der Liste wird blau dargestellt, der sortierte Bereich rot. Grün werden jene Elemente dargestellt, die gerade miteinander verglichen werden.

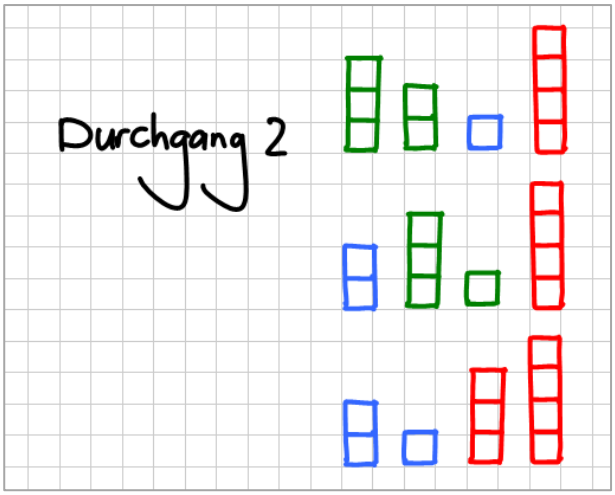


Durchgang 1: Der unsortierte Teil der Liste wird durchlaufen (im 1. Durchgang die gesamte Liste). Elemente, welche in der falschen Reihenfolge sind, werden vertauscht. Am Schluss ist das grösste Element ganz rechts und wird dem sortierten Bereich der Liste (rot markiert) zugeteilt.



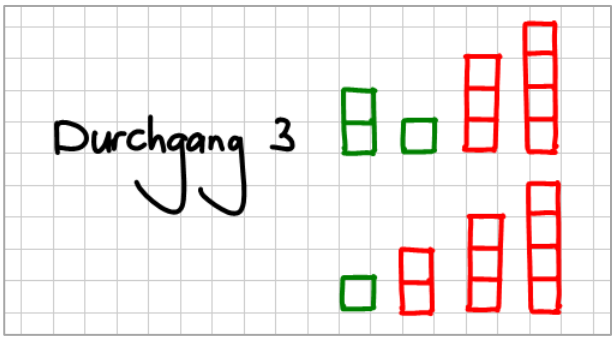
Anzahl Vertauschungen: 2

Durchgang 2: Wie beim ersten Durchgang wird der unsortierte Teil der Liste durchlaufen. Dieser enthält nun ein Element weniger. Auch diesmal werden falsch sortierte Elemente vertauscht. Am Schluss enthält der sortierte Bereich der Liste zwei Elemente.



Anzahl Vertauschungen: 2

Durchgang 3: Wie beim ersten und zweiten Durchgang wird der Bereich der unsortierten Liste durchlaufen.



Anzahl Vertauschungen: 1

Da der unsortierte Bereich der Liste am Schluss nur noch ein Element enthält, stoppt der Algorithmus. Wir haben eine sortierte Liste. Auch hier mussten insgesamt fünf Vertauschungen vorgenommen werden.

Die beiden beschriebenen Sortieralgorithmen gehören im allgemeinen Fall nicht zu den schnellstmöglichen. Es gibt eine Vielzahl von schnelleren, aber auch langsameren Sortieralgorithmen, die Sie bei Interesse unter https://de.wikipedia.org/wiki/Sortierverfahren finden. Diese lassen sich auch schön visualisieren, wie folgende Seite zeigt https://www.toptal.com/developers/sorting-algorithms.

2.4.3 Verschlüsselung

Ijr Ljmjnrsnx fzk ijw Xuzw! Nein, hier haben sich keine Tippfehler eingeschlichen. Die Nachricht ist aber verschlüsselt. Bestimmt finden Sie nach dem Lesen des Kapitels heraus, was es heisst.

Im Modul «Datenstrukturen im Zyklus 2» wurde die Thematik des Verschlüsselns bereits aufgegriffen. Hier soll vor allem der algorithmische Aspekt in den Vordergrund gerückt werden.

Geheime Botschaften zu verschicken, die nur der vorgesehene Empfänger entschlüsseln kann, ist seit jeher ein Bedürfnis. Die Kunst des Ver- und Entschlüsselns wird Kryptographie oder Kryptologie genannt. Bereits lange vor Christi Geburt tüftelten kluge Köpfe an Möglichkeiten. Ein Beispiel ist die «Skytale» (griech.: scytale; σκυτάλη, [sky'talə]: Stock, Stab), die vor rund 2500 Jahren zum Einsatz kam. Auf einem Papierstreifen stand zum Beispiel

S I H L T I T ADOC I U E L H S P R O ET T K G R D Z R I H A I Y E E S E P !

Dieser wurde um einen prismatischen Holzstab mit einer bestimmten Anzahl Kanten gewickelt. Die **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt einen solchen Stab mit sechs Kanten.

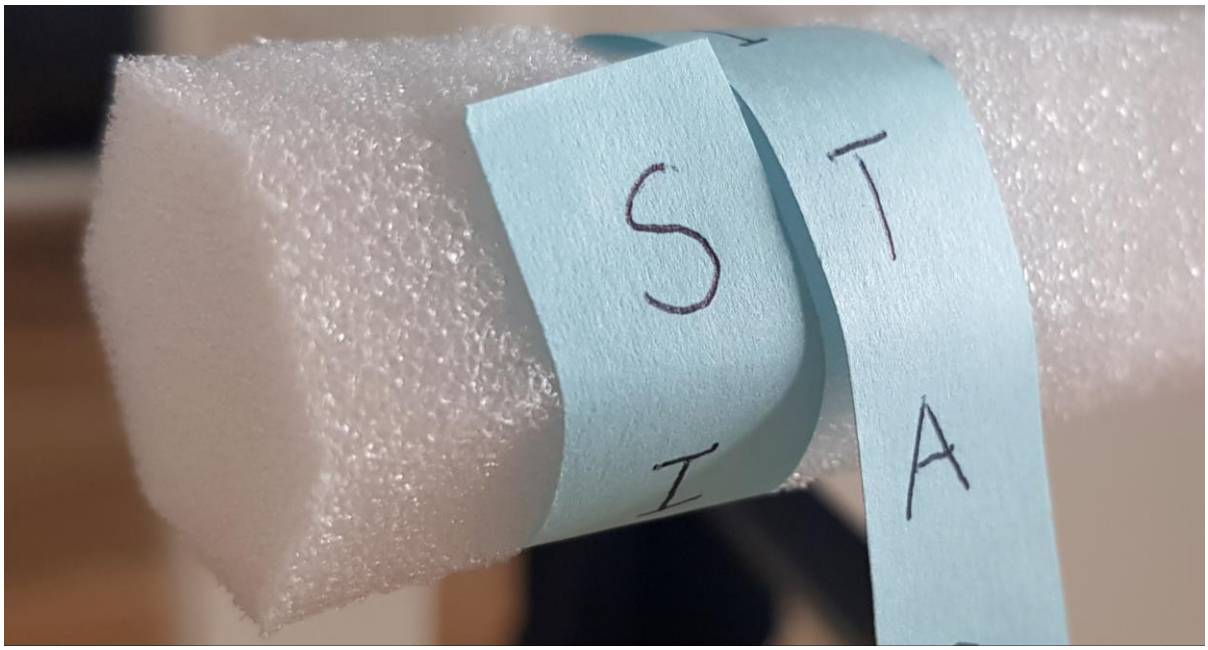


Abbildung 25 Geheime Botschaften auf einen Stab mit sechs Kanten aufgewickelt.

Wickelte man den obigen Papierstreifen um einen Stab mit dem Umfang von 4 Buchstaben, erhielt man

S T D U S E G R I S

I I O E P T R I Y E

H T C L R T D H E P

L A I H O K Z A E !

Wickelt man ihn aber um einen Stab mit dem Umfang von 5 Buchstaben, kam folgende geheime Botschaft zum Vorschein:

S I C H E R H E

I T I S T D A S

H A U P T Z I E

L D E R K R Y P

T O L O G I E !

Später, aber immer noch vor Christi Geburt, chiffrierte Cäsar Nachrichten, indem er im Alphabet Buchstaben um einen festgelegten Wert verschob. War dieser Verschiebungswert z.B. 3, wurde aus einem A ein D, aus dem B ein E etc. Ein einfaches, aber effektives Verfahren, das Cäsar zu vielen Siegen verholfen hat, weil geheime Angriffspläne übermittelt werden konnten. Der einleitende Text wurde mit diesem Verfahren verschlüsselt. Sie müssen nur noch herausfinden, um wie viele Buchstaben verschoben wurden!

Mit der Zeit wurden die Verfahren immer ausgeklügelter und komplexer. Bestimmt haben Sie schon vom Enigma (Verschlüsselungsmaschine, die im 2. Weltkrieg zum Einsatz kam und den Kriegsverlauf wesentlich prägte) oder dem RSA-Verfahren gehört, das noch heute in ähnlicher Form beim E-Banking Anwendung findet. Leider sprengen sowohl das Enigma wie auch das RSA-Verfahren für dieses Dossier den Rahmen.

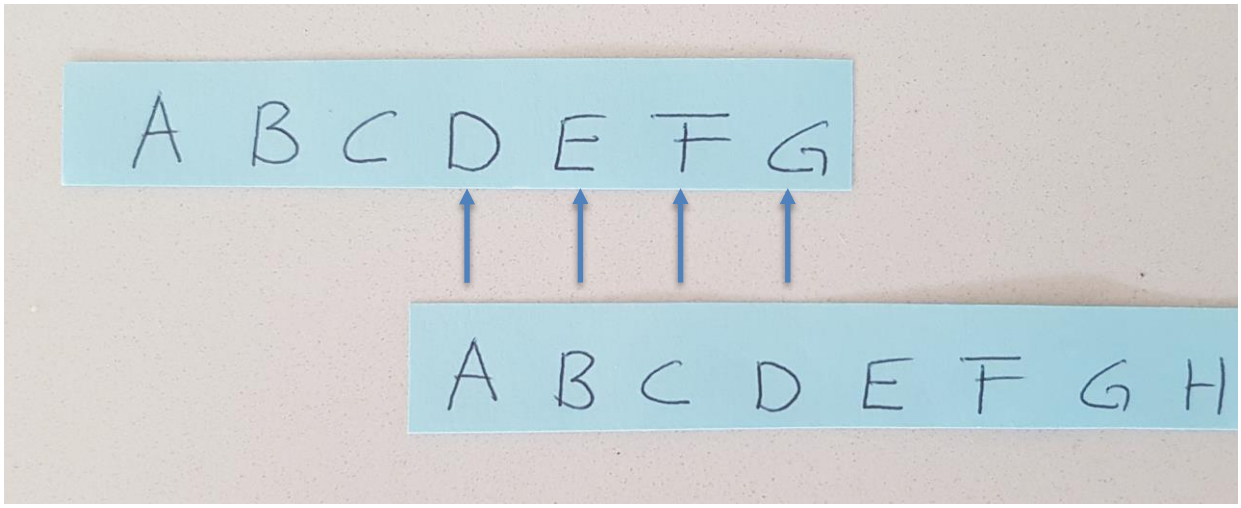


Abbildung 26 Caesar-Verschlüsselung (aus A wird D, …).

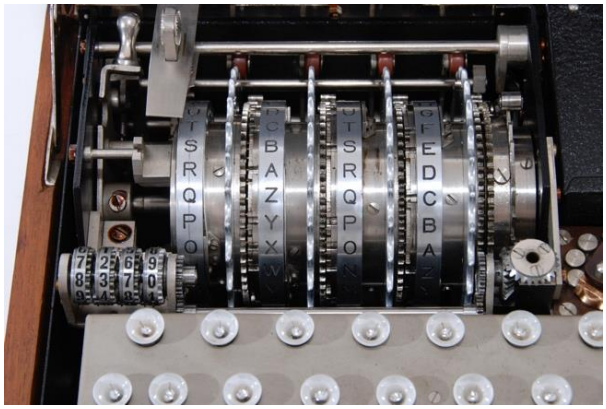


Abbildung 27 Enigma (cryptomuseum.com, 2017).

3 Fachdidaktischer Hintergrund

Im Grundlagenmodul sind bereits drei fachdidaktische Prinzipien wie «Selbständiges Entdecken», «Informatik ‚be-greifen'» und «Making» ausgeführt. Diese bilden Grundlagen und werden hier nicht weiter erläutert, kommen teilweise aber in Kombination mit anderen Prinzipien auch hier zum Tragen.

Wichtig für das vorliegende Modul «Algorithmen» ist das Prinzip des Hinterfragens. Fragen wie «Macht der Algorithmus wirklich das, was er soll?» oder «Kann die Anweisung für den Computer noch optimiert werden?» stehen im Zentrum. Ausgehend vom formulierten Text (Anweisungen), wird ein Struktogramm/Flussdiagramm erstellt und anschliessend in den Programm-Code übertragen und ausgeführt (siehe 2.3 «Programmieren»). Entscheidend ist, dass viele Möglichkeiten zum Ausprobieren geschaffen, die Anweisungen analysiert und angepasst sowie optimiert werden. Insbesondere der Analyse muss Platz eingeräumt werden. Konkret können verschiedene Anweisungen für Programme/Algorithmen zusammen verglichen, kommentiert und besprochen werden. So steht hier nicht nur «Es funktioniert!» im Zentrum, sondern auch «Warum funktioniert es?» und «Welches ist die bessere Lösung oder wie könnte es besser funktionieren?» (siehe auch Lehrplan 21 MI.2 2b).

Digitale Bildung lässt sich mit Hilfe des Dagstuhl-Dreiecks aus drei unterschiedlichen Perspektiven betrachten:

1. Technologische Perspektive:
   Wie funktioniert das?
2. Anwendungsorientierte Perspektive:
   Wie nutze ich das?
3. Gesellschaftlich-kulturelle Perspektive:
   Wie wirkt das?

Abbildung 28 Dagstuhl-Dreieck (GI-Deutsche Gesellschaft für Informatik, 2016).

Für eine vertiefte Auseinandersetzung mit der Fachdidaktik konsultieren Sie das Grundlagenmodul (MIA21-Grundlagenmodul). Im Folgenden werden zu einem ausgewählten Phänomen der informatischen Bildung gemäss Lehrplan 21 mögliche Aktivitäten und Anregungen aus den obigen drei Perspektiven aufgezeigt. Das unten aufgeführte Beispiel nimmt Bezug auf den Grundanspruch des Lehrplans 21 im Kompetenzbereich 2 (Algorithmen, MI.2.2f).

Tabelle 2: Einfache Problemstellung lösen durch Probieren.

<table>
<tr><th colspan="2">Einfache Problemstellungen lösen durch Probieren.</th></tr>
<tr><td>Technologische Perspektive</td><td>Fang mich!

Die Kinder programmieren mit «Scratch» z.B. ein Fang-mich-Spiel, bei welchem eine Katze einen Wollknäuel fangen soll. Der Wollknäuel bewegt sich zufällig nach oben, unten, links oder rechts und prallt am Rand ab. Die Katze wird vom Benutzer mit der Maus oder Tastatur gesteuert. Kommt die Katze bis zum Wollknäuel, bleibt dieser stehen und eine Siegesmelodie ertönt.

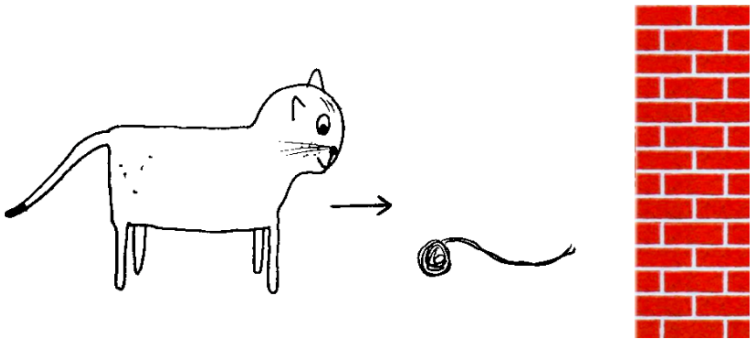
 

Abbildung 29 Katze will Wollknäuel fangen (Oehri, 2017).

Lehrplan 21: MI.2.2f: Die Schülerinnen und Schüler können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen.</td></tr>
<tr><td>Anwendungsorientierte Perspektive</td><td>Möchten Kinder an einem Getränkeautomaten ein Mineralwasser kaufen, wählen sie als Erstes das Produkt. Der Automat weiss, wie viel dies kostet (Parameter) und wartet in einer Schleife, bis der entsprechende oder grössere Betrag eingeworfen wurde. Sobald der Mindestbetrag vorhanden ist, gibt der Automat das Produkt frei und berechnet den Rückgabebetrag.

Andere Anwendungen am digitalen Gerät wären die Klassiker Tetris, PacMan etc.</td></tr>
<tr><td>Gesellschaftlich-kulturelle Perspektive</td><td>Unser Lebensalltag ist geprägt von Automaten, Geräten, Steuerungen und Prozessen, bei welchen Entscheidungen und Schleifen eine zentrale Rolle spielen. So kann zum Beispiel ein Spielzeug erst gekauft werden, wenn der nötige Betrag zusammengespart wurde, erst über die Strasse gelaufen werden, wenn die Ampel grün zeigt oder eine App erst gespielt werden, wenn der richtige Entsperrungscode eingegeben wurde.

Bei vielen Apps sind ähnliche Mechanismen vorgesehen. Beispielsweise kann man in «Clash of Clans» warten, bis Nahrung zur Verfügung steht oder man kann sie mit einem entsprechenden Geldbetrag unmittelbar freischalten (In-App-Kauf).</td></tr>
</table>

4 Praxisnahe Literatur mit Beispielen

Die Minibiber – Entdecke die Informatik! (ohne digitales Gerät)

www.minibiber.ch

«Die Minibiber» richten sich gemäss Lehrplan 21 primär an Lehrpersonen von Schülerinnen und Schülern im Zyklus 1. Die Aufgaben mit unterschiedlichem Schwierigkeitsgrad sind natürlich auch für den Zyklus 2 und 3 als knifflige Herausforderungen gedacht. Tauchen Sie ein in die spannende Welt der Minibiber und erleben Sie Informatik auf spielerischen und abenteuerlichen Wegen.

Der Informatikbiber (ohne digitales Gerät)

www.informatik-biber.ch

weckt das Interesse an Informatik durch spannende Aufgaben, die keine Vorkenntnisse erfordern.

zeigt jungen Menschen, wie vielseitig und alltagsrelevant Informatik ist.

regt zur weiteren Beschäftigung mit Informatik an.

ist ein Online-Wettbewerb, die Teilnahme daran dauert 40 Minuten.

Computer-Science unplugged (ohne digitales Gerät)

www.csunplugged.org

für Aktivitäten ohne digitales Gerät

führt in das «Computational Thinking» ein.

Primalogo (textbasiertes Programmieren)

www.primalogo.ch

Man lernt, ...

wie man für einfache geometrische Problemstellungen Lösungsstrategien (Algorithmen) entwirft und in Form eines Programms implementiert,

wie man komplexe Aufgabenstellungen durch modularen Entwurf auf einfachere Teilaufgaben zurückführt und

welche Konzepte der Steuerung des digitalen Gerätes zu Grunde liegen, in einer dem Alter der Schülerinnen und Schüler entsprechenden Form.

TigerJython (textbasiertes Programmieren)

www.tigerjython4kids.ch und www.tigerjython.ch

Man lernt, ...

wie man für einfache geometrische Problemstellungen Lösungsstrategien (Algorithmen) entwirft und in Form eines Programms implementiert,

wie man komplexe Aufgabenstellungen durch modularen Entwurf auf einfachere Teilaufgaben zurückführt und

welche Konzepte der Steuerung des Computers zu Grunde liegen, in einer dem Alter der Schülerinnen und Schülern entsprechenden Form.

Hour of Code – lerne Programmieren in einer Stunde (visuelles und textbasiertes Programmieren)

code.org/learn

Online-Programmieren mit visuellen Blöcken

Der Code wird aber auch in Textform präsentiert.

CodeCombat (textbasiertes Programmieren)

codecombat.com

Das beste Spiel, um Programmieren zu lernen.

Online-Programmieren mit textbasierter Eingabe

Lightbot – Online-Programmieren (visuelles Programmieren)

lightbot.com

Einfache Anweisungen durch Klicken

Fördert das Vorstellungsvermögen

Gute Einführung in das Programmieren (Anweisungen geben)

Scratch – Online- und Offline-Programmieren (visuelles Programmieren)

scratch.mit.edu

Erschaffe Geschichten, Spiele, und Animationen und teile sie mit anderen weltweit.

Einfache bis sehr komplexe Programmierung möglich

Snap – Erweiterung zu Scratch (visuelles Programmieren)

snap.berkeley.edu

Erweitertes Programmieren («for»-Schleife, Objekte, Rekursion, Rückgabe von Werten etc.)

Viele zusätzliche Erweiterungsmöglichkeiten (Lego, Arduino etc.)

AgentCubes – Coding for kids (visuelles Programmieren)

www.agentcubesonline.com

Online-Games programmieren in 3D.

Stellt hohe Anforderungen an die Abstraktion sowie die Bedienung des Programms.

Open-Roberta – Online Programmierumgebung (visuelles Programmieren)

lab.open-roberta.org

Simulation (Programmieren ohne Roboter)

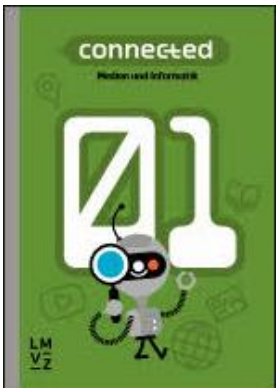
Einfache bis sehr komplexe Programmierung möglich

Open Roberta SiM (Ev3), Lego-EV3, Calliope (Microboard), Lego-NXT, Microbit etc., weitere in Entwicklung

Blocksystem, analog zu «Scratch»

Programmcode und Kommentarfunktion

connected 1 (Empfohlenes Lehrmittel Primarstufe)



Hartmann, W., Jurjevic, D., Senn, F., Waldvogel, B., & Zuberbühler, U. (2018). *connected 1*. Lehrmittelverlag des Kantons Zürich. *ISBN 978-3-03713-776-5*

(5. Klasse)

Die Kompetenzen im Modul «Medien und Informatik» des Lehrplans 21 werden in einer Wochenlektion während eines Schuljahrs abgedeckt. Handlungsorientierte Beispiele, die sich auch für integrativen Unterricht oder Projekte eignen. Für die Lehrpersonen steht ein digitales Handbuch bereit.

connected 2 (Empfohlenes Lehrmittel Primarstufe)

Hartmann, W., Jurjevic, D., Senn, F., Waldvogel, B., & Zuberbühler, U. (2019). connected 2. Lehrmittelverlag des Kantons Zürich. ISBN 978-3-03713-777-2

(6. Klasse)

Die Kompetenzen im Modul «Medien und Informatik» des Lehrplans 21 werden in einer Wochenlektion während eines Schuljahrs abgedeckt. Handlungsorientierte Beispiele, die sich auch für integrativen Unterricht oder Projekte eignen. Für die Lehrpersonen steht ein digitales Handbuch bereit.

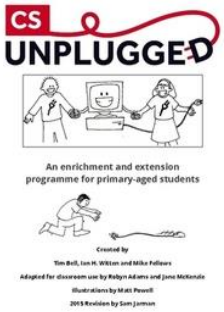
inform@21

Autorenteam (2017). *inform@21*. St. Gallen: Lehrmittelverlag St. Gallen. ISBN-Nr.: 978-3-905973-57-0

Mit Unterrichtsarrangements für die konkrete Umsetzung.

(5. und 6. Klasse)

Computer Science unplugged



Bell., T., Witten, Ian H., & Fellows, M. (2015). http://csunplugged.org

Die Aktivitätensammlung deckt viele Aspekte der Informatik ab, unter anderem die Bereiche «Datenstrukturen» und «Algorithmen». Die Webseite und das Buch sind in Englisch verfasst. Einzelne Aktivitäten wurden auf Deutsch übersetzt und sind auf der Webseite unten zu finden.

Das ganze Buch in Englisch kann auf der Webseite als PDF- oder Word-Dokument heruntergeladen oder in gedruckter Version bei lulu.com bestellt werden. Im Buchhandel ist das Buch nicht erhältlich.

Lernphase C: Umsetzung

1 Darum geht's

- Sie haben in der Lerngruppe ein eigenes Unterrichtsszenario erarbeitet und in Ihrem Unterricht umgesetzt und dokumentiert.
- Sie verfügen über eine Vielfalt von konkreten Unterrichtsideen zum Thema.

2 Vorgehen bei der Aufgabenbearbeitung

Ihre Aufgabe ist es nun, ein konkretes Unterrichtsszenario zu planen und zu beschreiben. Entscheiden Sie sich innerhalb der Lerngruppe für eine Aufgabenmöglichkeit, welche Sie folgendermassen bearbeiten:

4. Erstellen eines Entwurfs für ein Unterrichtsszenario gemäss Vorlage
   - Variante 1: *Vorlage MIA21 Lernphase3_Aufgabeneinreichung.docx*
   - Variante 2: *Vorlage der eigenen Pädagogischen Hochschule*
   Speichern Sie das Dokument mit folgender Beschriftung:
   Modulname_VornameNachname_JJJJMMTT.docx
   (Beispiel: *Informationsrecherche_PeterMuster_20160925.docx*).
   Reichen Sie die Aufgabe per E-Mail bei Ihrer Mentorin bzw. Ihrem Mentor ein.
5. Feedback durch Mentor oder Mentorin
6. Überarbeitung und Einreichung der überarbeiteten Version des Unterrichtsszenarios
7. Kurzfeedback
8. Durchführung im Unterricht
9. Reflexion des Unterrichts

Wählen und bearbeiten Sie eine der folgenden drei Aufgaben gemäss den oben beschriebenen Schritten 1 bis 6.

3 Aufgaben

3.1 Aufgabe A1: Programmieren

Mathematik mit «Scratch» und/oder «TigerJython»

Planen und beschreiben Sie ein Unterrichtsszenario für das Fach Mathematik. Dies soll Programmierkonzepte mit «Scratch» und/oder «TigerJython» thematisieren und kann mit oder ohne digitales Gerät ausgeführt werden. Wählen Sie dazu einen Inhalt/eine Kompetenz aus dem Lehrplan zum Fach Mathematik und überlegen Sie sich, wie dies mit einem der beiden Programmierkonzepte umgesetzt/erweitert/angereichert werden kann.

Halten Sie Ihre Planung im entsprechenden Planungsformular fest. Dabei sollen Ihre didaktischen Überlegungen wie Ziele, Sozialformen, zeitliche Planung, verwendete Medien etc. klar beschrieben sein.

Reichen Sie als Anlage zusätzlich alle Arbeitsblätter, Unterrichtsmaterialien und schriftlichen Anleitungen ein.

Ihre Beschreibung soll folgende Punkte abdecken:

- Bezug zum Fach Mathematik.
- Allenfalls Bezug zum Alltag.
- Angabe der Teilkompetenzen aus dem Lehrplan 21 sowohl aus dem Fach Mathematik wie auch aus dem Fach Medien und Informatik (Teilgebiet Informatik).
- Ausgewählte Aufgabenstellungen inklusive Begründung für den Einsatz im Unterricht.
- Berücksichtigung der didaktischen Funktionstypen der gewählten Aufgabenstellungen aus dem Grundlagenmodul MIA21 (Konfrontationsaufgabe/Erarbeitungsaufgabe/Übungs- und Vertiefungsaufgabe/Transfer- und Syntheseaufgabe/summative oder formative Beurteilungsaufgabe).
- Bestimmung des Kompetenzniveaus der gewählten Aufgabenstellungen.
- Organisation: Raumorganisation, Material, Sozialform.
- Beobachtungs- und Analysekriterien für die Lernenden.
- Innere Differenzierungsmöglichkeiten, die im Unterricht eingesetzt werden können.
- Überlegungen, wie eine altersgerechte Reflexion der Einheit durchgeführt werden kann.

3.2 Aufgabe A2: Programmieren

Bildnerisches Gestalten mit «Scratch» und/oder «TigerJython»

Planen und beschreiben Sie ein Unterrichtszenario für das Fach Bildnerisches Gestalten. Dies soll Programmierkonzepte mit «Scratch» und/oder «TigerJython» thematisieren und kann mit oder ohne digitales Gerät geschehen. Wählen Sie dazu einen Inhalt/eine Kompetenz aus dem Lehrplan zum Fach Gestalten und überlegen Sie sich, wie dieses mit einem der beiden Programmierkonzepte umgesetzt/erweitert/angereichert werden kann.

Halten Sie Ihre Planung im entsprechenden Planungsformular fest. Dabei sollen Ihre didaktischen Überlegungen wie Ziele, Sozialformen, zeitliche Planung, verwendete Medien etc. klar beschrieben sein.

Reichen Sie als Anlage zusätzlich alle Arbeitsblätter, Unterrichtsmaterialien und schriftlichen Anleitungen ein.

Ihre Beschreibung soll folgende Punkte abdecken:

- Bezug zum Fach Bildnerisches Gestalten.
- Allenfalls Bezug zum Alltag.
- Angabe der Teilkompetenzen aus dem Lehrplan 21 sowohl aus dem Fach Gestalten wie auch aus dem Fach Medien und Informatik (Teilgebiet Informatik).
- Ausgewählte Aufgabenstellungen inklusive Begründung für den Einsatz im Unterricht
- Berücksichtigung der didaktischen Funktionstypen der gewählten Aufgabenstellungen aus dem Grundlagenmodul MIA21 (Konfrontationsaufgabe/Erarbeitungsaufgabe/Übungs- und Vertiefungsaufgabe/Transfer- und Syntheseaufgabe/summative oder formative Beurteilungsaufgabe).
- Bestimmung des Kompetenzniveaus der gewählten Aufgabenstellungen.
- Organisation: Raumorganisation, Material, Sozialform.
- Beobachtungs- und Analysekriterien für die Lernenden.
- Innere Differenzierungsmöglichkeiten, die im Unterricht eingesetzt werden können.
- Überlegungen, wie eine altersgerechte Reflexion der Einheit durchgeführt werden kann.

3.3 Aufgabe A3: Selbst definierte Aufgabe

Wenn das Team möchte, können Sie die vorangehenden Aufgaben verändern oder kombinieren, sodass Sie eine Aufgabe mit konkreten Inhalten Ihrer eigenen Wahl erhalten. Die alternative Aufgabe muss von Ihrem Mentor bzw. Ihrer Mentorin genehmigt werden.

Planen und beschreiben Sie ein Unterrichtsszenario, in welchem die Schülerinnen und Schüler Informatische Bildung entdecken, erleben, anwenden und üben können. Dabei müssen …

- die Überlegungen zu informatischer Bildung, die Sie sowohl im Grundlagenmodul als auch im vorliegenden Modul kennengelernt haben, im Lernszenario ausreichend berücksichtigt sein.
- Sie die fundamentalen Ideen der informatischen Bildung sowohl hinsichtlich fachspezifischer als auch pädagogischer Gesichtspunkte ausreichend berücksichtigen und diskutieren (siehe Grundlagenmodul MIA21).
- Sie die didaktischen Funktionstypen der gewählten Aufgabenstellungen gemäss dem Grundlagenmodul kennen (Konfrontationsaufgabe/Erarbeitungsaufgabe/Übungs- und Vertiefungsaufgabe/Transfer- und Syntheseaufgabe/summative oder formative Beurteilungsaufgabe).
- die Schülerinnen und Schüler informatischen Fragen nachspüren bzw. einen Einblick in einfache technisch-informatische Zusammenhänge erhalten.
- das Ausmass und der Umfang der Aufgabe den vorangegangenen Aufgaben entsprechen.

Halten Sie Ihre Planung im entsprechenden Planungsformular fest. Dabei sollen Ihre didaktischen Überlegungen wie Ziele, Sozialformen, zeitliche Planung, verwendete Medien etc. klar beschrieben sein. Reichen Sie als Anlage zusätzlich alle Arbeitsblätter, Unterrichtsmaterialien und schriftlichen Anleitungen ein.

Lernphase D: Abschluss und Reflexion

1 Darum geht's

- Sie haben auf Ihren Lernprozess in diesem bearbeiteten Modul zurückgeschaut und Ihre Erkenntnisse schriftlich festgehalten.

2 Persönliche Reflexion

Schauen Sie auf Ihren Lernprozess während des Moduls zurück und dokumentieren Sie Ihre Erkenntnisse anhand folgender Fragestellungen. Stellen Sie Ihre Dokumentation des Lernprozesses als Abschluss des Moduls Ihrem Mentor/Ihrer Mentorin zu.

1. Ebene Unterricht

- Was hat sich in Ihrer Planung bewährt, was mussten Sie ändern?
- Wie beurteilen Sie den Lernzuwachs in Bezug auf die Kompetenzen des Lehrplans 21, «Algorithmen» (siehe Lernphase A dieses Moduls) Ihrer Schülerinnen und Schüler?
- Wie werden Sie in diesem Kompetenzbereich weiterfahren?

2. Ebene persönlicher Lerngewinn

- Gehen Sie in Gedanken nochmals zurück an den Start des Moduls: Welche Kompetenzen haben Sie in Bezug auf Informatik, Kompetenzbereich «Algorithmen» dazugewonnen?
- Wie haben Sie den Lernprozess in der Lerngruppe erlebt?
- Inwiefern hat sich die Auseinandersetzung im Modul auf Ihren Unterricht ausgewirkt?
- Wie beurteilen Sie die Arbeit mit diesem Modul?

Hintergrundwissen und weitere Literatur

Sie möchten sich weiter ins Thema vertiefen? Gerne empfehlen wir Ihnen folgende Literatur:

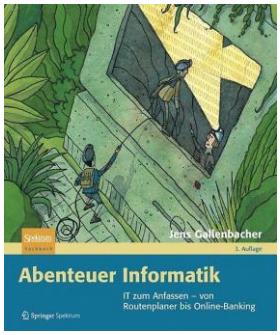
Spielend programmieren lernen

Wainerwright, M., Henson, M., & Klocker, U. (2016). *Spielend programmieren lernen.* Ravensburger-Verlag. ISBN-Nr.: 978-3-473-55436-2

Informatik Programmieren

(How to Code 1–4 → Originaltitel)

Abenteuer Informatik



Gallensbacher, J. (2012). *Abenteuer Informatik.* Heidelberg: Springer Spektrum. ISBN-Nr.: 978-3-8274-2965-0

Informatik ohne Computer oder digitales Gerät

IT zum Anfassen – von Routenplaner bis Online-Banking

Programmieren super easy

Dorling Kindersley (2014). *Programmieren super easy*. München: Dorling Kindersley. ISBN-Nr.: 978-3-8310-2700-2

Informatik programmieren

Einfacher Einstieg in «Scratch» und «Phyton»

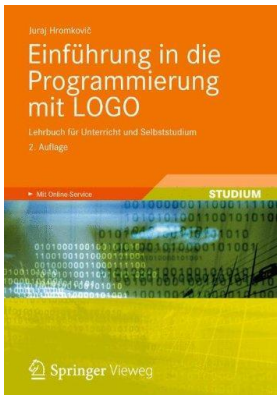
Scratch 2.0 – Spielend programmieren lernen

Ullwer J. (2015). *Scratch 2.0 – Spielend programmieren lernen*. Bodenheim: Herd-Verlag. ISBN-Nr.: 978-3-86249-367-8

Informatik programmieren

Lehrerband und Arbeitsheft

Einführung in die Programmierung mit LOGO



Hromkovic, J. (2012). *Einführung in die Programmierung mit LOGO*. Heidelberg: Springer Vieweg. ISBN-Nr.: 978-3-8348-1852-2

Informatik Programmieren

Lehrbuch für den Unterricht und das Selbststudium

Didaktik der Informatik

Hubwieser, P. (2007). *Didaktik der Informatik*. Heidelberg: Springer Verlag. ISBN-Nr.: 978-3-540-72477-3

Didaktik der Informatik

Das Werk bietet ein schlüssiges Gesamtkonzept für die Didaktik der Informatik, angefangen bei lernpsychologischen Grundlagen über allgemeine didaktische Prinzipien hin zu Hinweisen für die Unterrichtsplanung in der Praxis. Einige der Praxisbeispiele sind allerdings zu anspruchsvoll für die Primarschule.

Literaturverzeichnis

code.org. (20. 06 2019). code.org. Von code.org: https://code.org/ abgerufen

cryptomuseum.com. (23. 01 2017). *cryptomuseum.com.* Abgerufen am 23. 01 2017 von cryptomuseum.com: http://cryptomuseum.com/crypto/enigma/img/300166/035/full.jpg

Doebeli, B. (15. 10 2015). *http://beat.doebe.li/.* Abgerufen am 24. 01 2017 von http://beat.doebe.li/: http://beat.doebe.li/

Gabler Springler. (08. 12 2016). *http://wirtschaftslexikon.gabler.de*. (S. G. Verlag, Herausgeber) Abgerufen am 08. 12 2016 von http://wirtschaftslexikon.gabler.de: http://wirtschaftslexikon.gabler.de/Archiv/57779/algorithmus-v9.html

GI-Deutsche Gesellschaft für Informatik. (15. 10 2016). *www.gi.de.* Abgerufen am 27. 01 2017 von www.gi.de: https://www.gi.de/aktuelles/meldungen/detailansicht/article/dagstuhl-erklaerung-bildung-in-der-digitalen-vernetzten-welt.html

huehner-info.de. (17. 10 2016). *www.huehner-info.de.* Abgerufen am 17. 10 2016 von www.huehner-info.de: http://www.huehner-info.de/images/input_output.jpg

media.kswillisau.ch. (23. 01 2017). *media.kswillisau.ch.* Abgerufen am 23. 01 2017 von media.kswillisau.ch: http://media.kswillisau.ch/in/procStart/index.html

Oehri, E. (6. Januar 2017). MINT unterwegs - Projekt DVS Luzern. *MINT unterwegs - Projekt DVS Luzern*. Luzern, LU, CH.

PH Luzern. (09. 12 2016). Zembi PH Luzern Einführung LP21 Informatik Z2. *Zembi PH Luzern Einführung LP21 Informatik Z2*. Luzern, Luzern, CH: ZEMBI PH Luzern.

spektrum.de. (08. 12 2016). *www.spektrum.de*. Abgerufen am 08. 12 2016 von www.spektrum.de: http://www.spektrum.de/fm/912/thumbnails/423405.gif.815439.gif

Volksschule Buchrain, B. (08. 12 2016). *buchrain.educanet2.ch*. Abgerufen am 08. 12 2016 von buchrain.educanet2.ch: http://buchrain.educanet2.ch/pt11/.ws_gen/17/DSC01121.JPG

wikimedia.org. (23. 01 2019). *wikimedia.org.* Abgerufen am 23. 01 2017 von wikimedia.org: https://upload.wikimedia.org/wikipedia/commons/7/75/Caesar_substition_cipher.png

wikipedia.org. (23. 01 2017). *wikipedia.org*. Abgerufen am 23. 01 2017 von wikipedia.org: https://de.wikipedia.org/wiki/Skytale

wikipedia.org. (27. 01 2017). *wikipedia.org*. Abgerufen am 27. 01 2017 von wikipedia.org: https://de.wikipedia.org/wiki/Bin%C3%A4re_Suche

www.origami-kunst.de. (05. 01 2017). *www.origami-kunst.de.* Abgerufen am 05. 01 2017 von www.origami-kunst.de: http://www.origami-kunst.de/faltanleitungen/diagramme/himmel-hoelle/

1 Abbildungsverzeichnis

2 Tabellenverzeichnis