Handreichung zum Lehrplan Informatik

der Oberschule

für die Klassenstufen 7 und 8

Ergänzung zum IMPRESSUM

Die vorliegende Fassung des Arbeitspapieres stellt eine Überarbeitung der Ausgabe von 2004 dar. Wesentliche Änderungen an den Lehrplänen wurden berücksichtigt, in Details und Formulierungen wurde das Papier in den meisten Teilen auf dem Stand von 2004 belassen. Bitte berücksichtigen Sie dies bei der Nutzung des Materials.

Überarbeitung: Nico Albanis

Redaktion: Dr. Holger Rohland, Thomas Knapp

Januar 2018

IMPRESSUM

Diese Handreichung hat den Charakter eines Arbeitspapiers.

Dieses Material wurde in einem Workshop an der AVS Meißen erarbeitet von:
Helmar Fischer, Heiko Neupert, Kerstin Redetzky, Olaf Schaarschmidt, Lutz Seidel, Frank Tannhäuser, Jens Walsch

Des Weiteren wurden Ergebnisse von Projektarbeiten, die im Lehramtsstudium Informatik an der TU Dresden im Wintersemester 2003/2004 entstanden, genutzt.
Beteiligt waren:
Katrin Christmann, Ellen Engelmann, Frank Götze, Angelika Kloos, Mario Krömer, Ralf Kügler, Kerstin Redetzky, Birgit und Uwe Schenk, Angela Vogel, Uwe Weise, Michael Wötzel,

Außerdem waren beteiligt:
Birgit Carstens, Karin Churs, Robby Buttke, Thomas Dittrich, Steffen Friedrich, Thomas Knapp, Birgit Langer, Bettina Timmermann

Redaktion: Heiko Neupert, Thomas Knapp

Layout: Helmar Fischer

Die Veröffentlichung erfolgt durch:
Technische Universität Dresden, Fakultät Informatik
Professur Didaktik der Informatik
01062 DRESDEN

Hinweise, Ergänzungen und Kritiken werden gern gesehen und sollten an die obige Adresse geschickt werden.

Dresden, Juli 2004

1 Einige wichtige Vorbemerkungen

Das Erscheinen neuer Lehrpläne bringt Wohl und Wehe und manch hitzige Debatte. Vor allem aber braucht es eine intensive Anstrengung aller Kolleginnen und Kollegen, um die neuen Ideen in neuem Unterricht wirklich realisieren zu können. Diesem Grundgedanken folgend ist diese Broschüre entstanden.

Das Unterrichtsfach Informatik an den sächsischen Oberschulen steht inzwischen in einer guten Tradition und hat sich in den zurückliegenden mehr als 10 Jahren einen anerkannten Stand an den meisten Schulen erworben. Ausgehend von den kaum noch zu übersehenden Entwicklungen zur Informations- und Wissensgesellschaft in vielen Bereichen des täglichen Lebens werden grundlegende Kompetenzen zur Beherrschung informationeller Prozesse wichtiger denn je. Dass dafür die Bedienung gerade aktueller Anwendungen kaum ausreichend ist, wurde durch das Festhalten und der stabilen Einordnung des Unterrichtsfaches Informatik im Mittelschulkonzept unterstrichen.

Neue Lehrpläne fordern allerdings auch eine gewisse Vorausschau auf künftige Entwicklungen, ein Benennen der zur Bewältigung künftiger Anforderungssituationen notwendigen Voraussetzungen. Dabei haben die fachlichen Grundlagen, die aus Sicht der Informatik den allgemeinbildenden Kern darstellen, eine stabile Ausprägung erreicht, sofern man eine Abkopplung von gerade aktueller Software vornimmt. Dem folgt der neue didaktische Ansatz des Lehrplans, der den zu nutzenden Applikationen den Status des Exemplarischen gibt. Das jeweilige Anwendungsprogramm wird nicht seiner selbst willen, sondern zur Vermittlung informatischer Sachverhalte eingesetzt. Die Entscheidung dazu wird nicht mehr durch den Lehrplan zentralisiert, sondern in die Verantwortung der Fachlehrer und Lehrerkonferenzen übergeben. Gerade letztere Gremien sind notwendig, um schulspezifischen Informatikunterricht zu erteilen und fachübergreifend die Computernutzung in den Fächern zu koordinieren. Stärker als bisher ist vom Informatikunterricht an Oberschulen deshalb auch zu erwarten, dass unter Beachtung einer altersspezifischen didaktischen Reduzierung grundlegende Vorgehensweisen der Informatik, zum Beispiel Abstraktion und Modellierung, eine Rolle spielen.

Durch dieses Herangehen werden sehr viele Lehrerinnen und Lehrer kurzfristig vor der Aufgabe stehen, ihre Unterrichtskonzeption für das nächste Schuljahr grundlegend zu überarbeiten. Sicher wird im Vergleich mit dem bisherigen Lehrplan sehr schnell festgestellt werden, dass den häufig gestellten Forderungen nach einer systematischen Einführung in begriffliche und technische Grundlagen nun besser entsprochen wurde. Vor allem wird eine größere Freiheit zur eigenen Gestaltung des Unterrichts geboten. Es muss an dieser Stelle nicht ausdrücklich betont werden, welche Funktion Lehrpläne besitzen. Das vorliegende Material kann lediglich eine Unterstützung bieten und helfen, die fixierten Anforderungen für den eigenen Unterricht umzusetzen.

Insbesondere wurde versucht, aus didaktischer Sicht Schwerpunkte zu setzen und – über die Darstellung der Ziele und deren Progression in den weiteren Klassenstufen – den Zusammenhang von Zielen und Inhalten im Lehrplan insgesamt zu verdeutlichen. Aus den dargestellten Zielen und den Möglichkeiten der zu deren Umsetzung verwendbaren Anwendungen sind verschiedene Varianten entstanden. Sie sind als möglicher Gang durch den Unterricht detailliert beschrieben, und sind weder vollständig noch die einzig mögliche Vorgehensweise zur Realisierung des Lehrplanes. Eine weitere Orientierung für das anzustrebende Anforderungsniveau bilden ausgewählte Begriffe, die in einer schülergerechten Formulierung angegeben sind. Das gilt in gleicher Weise für die Zusammenstellung von Aufgaben, die den einzelnen Lernbereichen zugeordnet wurden. Es erscheint vernünftig, die Auswahl und Modifikation von Aufgaben dem Lehrenden vollständig zu überlassen. Die Darstellung weiterer Fachbegriffe und Erläuterung für den Lehrer ist nur exemplarisch und erfordert die vertiefende Beschäftigung mit weiteren Quellen.

In diesem Sinne ist die vorliegende Handreichung natürlich nur ein Angebot für Informatik unterrichtende Lehrer und soll bei der Vorbereitung und Durchführung des Unterrichts in den Klassenstufen 7 und 8 der Oberschule helfen. Sie ist ein Anfang und durch Engagement von Kollegen aus den verschiedensten Teilen Sachsens in einer extrem kurzen Zeit entstanden. Im Vordergrund stand dabei die Idee, bereits vor Schuljahresbeginn ein unterstützendes Material zur Umsetzung des neuen Lehrplans vorzulegen.

An der Erstellung der Handreichung waren beteiligt:

Studierende im berufsbegleitenden Lehramtstudium an der TU Dresden, die in zusätzlicher Arbeit Inhalte aufbereitet haben

und

Lehrer, die freiwillig und auf eigene Kosten an einem Workshop teilgenommen und dort den Grundstock gelegt haben.

Letztlich gab es Einige, die unermüdlich das Ganze zu Ende gebracht haben.

Dieses Material ist weder ein Lehrbuch, noch eine Erläuterung von Unterrichtseinheiten. Es ist ein Arbeitsmittel, das anregen soll, den Informatikunterricht an den sächsischen Oberschulen im Geiste der Lehrpläne und zum Vorteil für die Schüler zu gestalten. Der konkrete Unterricht, die gesammelten Erfahrungen, die vielen weiteren Ideen – dies alles wird dieses Material vervollständigen, verbessern und erweitern.

Bei den Quellenangaben wird zwischen Druckerzeugnisse aller Art (Printquellen) und Quellen aus WWW (Webquellen) unterschieden. Sie sind ausführlich und nach Art getrennt im Kapitel 7 angegeben.

Wenn unsere Arbeit zum gemeinsamen Wirken für Informatikunterricht an den Oberschulen Sachsens anregt und recht viele Ideen diskutiert sowie Kritiken vorgebracht werden, wäre ein wichtiges Anliegen unserer Arbeit verwirklicht.

Dresden, Juli 2004

2 Allgemeine Informationen zum Lehrplan

2.1 Zielebenen des Lehrplans

Der Aufbau und die einzelnen Elemente des Lehrplans (Web: [1]) sind im Lehrplanmodell (Web: [2]) dargestellt. Die folgende Zusammenstellung verdeutlicht die Entwicklung der allgemeinen fachlichen Ziele und den damit verbundenen Anforderungen in den Jahrgangsstufenzielen im gesamten Informatikunterricht der Oberschule. Die Beachtung dieser Entwicklungslinie ist für eine ganzheitliche Vorgehensweise des Faches Informatik unerlässlich und sie erfordert eine jahrgangsübergreifende Planung. Die allgemeinen fachlichen Ziele und die Jahrgangsstufenziele werden in den Lernbereichen durch die konkreten Lernzielebenen untersetzt. Diese Lernzielebenen sind nach Lehrplanmodell einheitlich für alle Schularten und Fächer festgelegt. Sie unterscheiden sich jedoch in ihrer altersspezifischen, bildungsgangbezogenen und fachlichen Tiefe. Es ist also bei der persönlichen Planung zu beachten, dass auch gleiche Zielebenen entsprechend des Alters, des Bildungsgangs und des Inhalts unterschiedliche Konsequenzen für den Unterricht nach sich ziehen.

Im Folgenden wird an einigen Beispielen aus dem Lehrplan die unterschiedliche Ausprägung von Lernzielebenen verdeutlicht:

EINBLICK GEWINNEN zu Auswirkungen der Rechentechnik aus historischer Sicht (Klasse 7):

> An ausgewählten Beispielen erfahren die Schüler Zusammenhänge zwischen gesellschaftlicher Entwicklung und Entwicklung der Rechentechnik.

EINBLICK GEWINNEN in die gemeinsame Nutzung von Ressourcen in einem Netzwerk (Klasse 9):

> Die Schüler bearbeiten eine gemeinsame Datenbasis im Netz und werten diese aus. Sie wiederholen das Wissen zu Netzlaufwerken aus Klasse 7 und erweitern ihre Grundkenntnisse.

BEHERRSCHEN des Speicherns in einer Verzeichnisstruktur (Klasse 7):

> Die Schüler können Dateien aus verschiedenen Anwendungen sicher in Verzeichnisstrukturen speichern. Sie sind in der Lage, logische Zuordnungen von Dateien in Verzeichnisse zu erkennen

BEHERRSCHEN des gleichzeitigen Arbeitens mit verschiedenen Anwendungen (Klasse 10):

> Durch ihre Kenntnisse zum Client-Server-Prinzip können die Schüler selbständig und korrekt mit fremden Objekten in verschiedenen Anwendungen arbeiten. Sie nutzen die Methoden der Klassen in der jeweiligen Anwendung sicher.

Beherrschen in Klassenstufe 7 bedeutet somit ein vorrangig selbstständiges Arbeiten in überschaubaren Abschnitten, während in Klassenstufe 10 ein hoher Grad an Selbstständigkeit über einzelne Abschnitte erwartet wird.

Es sei an dieser Stelle nochmals betont: Diese Untersetzung der Zielebenen erfolgte exemplarisch und bildet in keiner Weise die Inhalte und Teilziele aus dem jeweiligen Lernbereich vollständig ab. Insbesondere ist zu beachten, dass aus jeder der Zielebenen entsprechend der Alterstufe und bezüglich eines konkreten Inhalts unterschiedliche Konsequenzen abzuleiten sind. Aus den Jahrgangszielen und deren Progression über die Klassenstufen sollte der inhaltliche Anspruch im Informatikunterricht erkennbar werden.

2.2 Zusammenstellung aller Jahrgangsstufenziele nach den allgemeinen fachlichen Zielen

Ausgehend von den Zielen und Aufgaben des Faches Informatik an Oberschulen sind allgemeine fachliche Ziele abgeleitet und für die einzelnen Klassenstufen untersetzt worden. Die zusammenfassende Darstellung ermöglicht einen Überblick über die Entwicklung dieser Ziele im gesamten Informatikunterricht.

A Aneignen von Strategien und Methoden des Umgangs mit Informationen und Daten

7	Die Schüler lösen typische Aufgaben zum Strukturieren von Daten und nutzen dabei ihre Kenntnisse aus anderen Fächern. Sie stellen Daten und Strukturen mit Hilfe geeigneter Modelle dar.
8	Die Schüler lösen typische Aufgabenklassen mit Hilfe ihrer Kenntnisse zu Modellen. Sie verwenden einfache Methoden der Bearbeitung von Informationen.
9	Die Schüler erweitern ihre methodischen Fähigkeiten zur Projektarbeit durch die Nutzung von Werkzeugen der Informatik.
10	Die Schüler festigen ihre Fähigkeit zur Teamarbeit beim selbstständigen und kreativen Bearbeiten komplexer Problemstellungen mit informatischen Mitteln.

B Nutzen von Informatiksystemen und Auseinandersetzen mit deren Wirkung auf Individuum und Gesellschaft

7	Die Schüler lösen ähnliche Aufgaben mit unterschiedlichen Programmen der gleichen Anwendung. Sie erkennen, wie die Nutzung des Computers ihre schulische Arbeit verändert.
8	Die Schüler lernen eine weitere Anwendung nutzen und erkennen deren Vorteile bei der Lösung einer neuen Aufgabenklasse. Sie erkennen, wie die Nutzung des Computers das gesellschaftliche Umfeld verändert.
9	Die Schüler nutzen eine Anwendung zum Verwalten von großen Datenmengen in gemeinsamen Ressourcen. Sie planen ihre Arbeit und reflektieren sie hinsichtlich der Zielerreichung. Die Schüler erfahren an ausgewählten Beispielen Grenzen der verwendeten Informatiksysteme.
10	Die Schüler nutzen ihr Wissen zielgerichtet zur Orientierung in unbekannten Systemen. Sie berücksichtigen Grenzen der verwendeten Informatiksysteme.

C Verwenden von informatischen Modellen und Modellierungstechniken

7	Die Schüler lernen erste Modelle kennen. Sie bedienen sich dieser Modelle, um sich in ein gegebenes Informatiksystem einzuarbeiten.
8	Die Schüler nutzen Modelle zielgerichtet. Sie gebrauchen Fachbegriffe sicher und können diese in die Fachsystematik einordnen.
9	Die Schüler erweitern ihre Kenntnisse zu Modellen und Modellierungsmethoden und verwenden diese bewusst.
10	Die Schüler vertiefen ihre Kenntnisse zu Modellen und Modellierungsmethoden bei der selbstständigen Arbeit an neuen Sachverhalten.

D Nutzen von Problemlösestrategien

7	Die Schüler vollziehen einzelne Schritte des Problemlöseprozesses an einfachen Beispielen unter Verwendung bekannter Werkzeuge nach.
8	Die Schüler modellieren und implementieren Lösungen zu einfachen Problemen.
9	Die Schüler modellieren und implementieren Lösungen zu komplexen Problemen.
10	Die Schüler modellieren und implementieren Lösungen zu komplexen Problemen. Sie wählen dazu notwendige Werkzeuge zielgerichtet aus.

Diese Zusammenstellung der Ziele zeigt die Umsetzung der Anforderung zu Wissensaneignung, Methodenkompetenz und Werteerziehung in den jeweiligen Klassenstufen.
Deutlich ablesbar ist einerseits die Steigerung des Anforderungsniveaus bei der Aneignung von Wissen und Können innerhalb eines allgemeinen fachlichen Zieles. Andererseits kann die zielgerichtete Entwicklung von Fähigkeiten und Fertigkeiten jahrgangsübergreifend geplant werden. Auf dieser Basis kann weiterhin fachübergreifend und fächerverbindend an der Herausbildung von Haltungen und Einstellungen systematisch gearbeitet werden.

2.3 Progression in den Lernzielen und Lerninhalten

Zur differenzierten Einschätzung der zu erwartenden Voraussetzungen und des anzustrebenden Niveaus im Rahmen des Unterrichts Technik und Computer in Klasse 5/6 (Web: [3]) und im Informatikunterricht in den Klassen 7 und 8 sollen die folgenden Tabellen eine Unterstützung bieten. Sicherlich können diese Übersichten nicht differenzierte Überlegungen zur eignen Planung ersetzen. Sie sollten dazu anregen, das dargestellte Niveau durch geeignete Aufgaben zu überprüfen.
Die Tabellen orientieren sich an den bereits dargestellten Jahrgangszielen.

Aneignen von Strategien und Methoden des Umgangs mit Informationen und Daten

Abschlussniveau 6 Technik und Computer	**Klassenstufe 7**	**Klassenstufe 8**
Kennen von Begriffen für die Arbeit mit dem Computer (5)	Kennen des prinzipiellen Aufbaus eines Computers Gestalten von Verzeichnisstrukturen auf dem Computer Beherrschen des Speicherns in einer Verzeichnisstruktur	Kennen grundlegender Programmstrukturen Beherrschen der Umsetzung des Modells an einfachen Beispielen
Einblick gewinnen in das Darstellen von Informationen mit einer Textverarbeitungssoftware (5)	Kennen grundlegender Datenstrukturen in einer ausgewählten Anwendung	Kennen von Klassen Beherrschen der Zuordnung zwischen Objekten und Klassen Gestalten einer Präsentation zu einer bekannten Persönlichkeit der Informatik / Mathematik (WP 3)
Einblick gewinnen in Möglichkeiten der Informationsbeschaffung mit computergestützten Medien (5)	Kennen ausgewählter Codes und Chiffren unter historischem Aspekt (WP 2)	Einblick gewinnen in das maschinelle Codieren und Chiffrieren von Texten mithilfe von Algorithmen (Wp 2)
Anwenden einer Form der elektronischen Kommunikation zum gemeinsamen Arbeiten (6)	Anwenden eines Codes auf das Codieren und Decodieren und einer Chiffre auf das Chiffrieren und Dechiffrieren einfacher Botschaften (WP 2)	

Nutzen von Informatiksystemen und Auseinandersetzen mit deren Wirkung auf Individuum und Gesellschaft

Abschlussniveau 6 Technik und Computer	**Klassenstufe 7**	**Klassenstufe 8**
Kennen von Begriffen für die Arbeit mit dem Computer (5)	Kennen des prinzipiellen Aufbaus eines Computers Übertragen des Prinzips „Eingabe-Verarbeitung-Ausgabe“ auf Vorgänge im Alltag	Kennen grundlegender Programmstrukturen Beherrschen der Umsetzung des Modells an einfachen Beispielen
Einblick gewinnen in das Darstellen von Informationen mit einer Textverarbeitungssoftware (5)	Gestalten von Verzeichnisstrukturen auf dem Computer Kennen grundlegender Datenstrukturen in einer ausgewählten Anwendung	Kennen von Klassen Beherrschen der Zuordnung zwischen Objekten und Klassen
Einblick gewinnen in Möglichkeiten der Informationsbeschaffung mit computergestützten Medien (5)	Beurteilen der Leistung eines Computers an verschiedenen Kriterien (WP3) Einblick gewinnen zu Auswirkungen der Rechentechnik aus historischer Sicht	Kritische Bewertung der Resultate des Problemlöseprozesses Übertragen der Kenntnisse zum Problemlöseprozess auf - selbständiges Lösen einfacher Probleme - kritische Bewertung der Resultate
Anwenden einer Form der elektronischen Kommunikation zum gemeinsamen Arbeiten (6)	Einblick gewinnen in Spielarten (WP1)	

Einblick gewinnen in Bestandteile des Computerarbeitsplatzes und deren Zusammenwirken (5)		
Beherrschen folgender Tätigkeiten beim Arbeiten mit dem System der Schule anhand einer gewählten Anwendung: - Herstellen der Systembereitschaft - Bedienen der Benutzeroberfläche - Eingeben und Bearbeiten von Daten - Speichern und Öffnen von Dateien (5)		
Einblick gewinnen in Hilfesysteme (5)		
Übertragen der Kenntnisse auf die Erstellung eines Dokuments mit dem Computer (6)		
Einblick gewinnen in weitere Gesaltungsmöglichkeiten von Dokumenten (6)		

Verwenden von informatischen Modellen und Modellierungstechniken

Abschlussniveau 6 Technik und Computer	**Klassenstufe 7**	**Klassenstufe 8**
Kennen von Begriffen für die Arbeit mit dem Computer (5)	Übertragen des Prinzips „Eingabe-Verarbeitung-Ausgabe“ auf Vorgänge im Alltag Gestalten von Verzeichnisstrukturen auf dem Computer Beherrschen des Speicherns in einer Verzeichnisstruktur	Kennen grundlegender Programmstrukturen Beherrschen der Umsetzung des Modells an einfachen Beispielen
Einblick gewinnen in das Darstellen von Informationen mit einer Textverarbeitungssoftware (5)	Kennen grundlegender Datenstrukturen in einer ausgewählten Anwendung Beherrschen der Arbeit mit konkreten Objekten beim Lösen typischer Aufgaben mit der gewählten Anwendung unter medienerzieherischen Aspekten	Kennen von Klassen Beherrschen der Zuordnung zwischen Objekten und Klassen
	Übertragen der Kenntnisse zur Beuntzeroberfläche auf die Oberfläche von Spielen (WP1)	Anwenden der Kenntnisse zu Objekten auf Simulationsspiele (WP1)

Nutzen von Problemlösestrategien

Abschlussniveau 6 Technik und Computer	Klassenstufe 7	Klassenstufe 8
Einblick gewinnen in Möglichkeiten der Informationsbeschaffung mit computergestützten Medien (5)	Beherrschen der Arbeit mit konkreten Objekten beim Lösen typischer Aufgaben mit der gewählten Anwendung unter medienerzieherischen Aspekten Übertragen der Kenntnisse zur Objektorientierung auf - Lösen eines einfachen Problems - Arbeiten mit einem anderen Programm der gleichen Anwendung	Beherrschen der Zuordnung zwischen Objekten und Klassen Beherrschen der Umsetzung des Modells an einfachen Beispielen Kennen des Problemlöseprozesses Übertragen der Kenntnisse zum Problemlöseprozess auf selbstständiges Lösen einfacher Probleme
Einlick gewinnen in die Entwicklung der Nachrichtenübermittlung (5) (WP 4)		

Diese Übersichten sollen dem Informatiklehrer bei der Betrachtung des Lehrplanes aus unterschiedlichen Sichten sowie bei Ihrer persönlichen Planung inhaltlich wie auch didaktisch-methodisch helfen. Die Zuordnung zu den Zielebenen bildet dabei lediglich ein Orientierungskriterium dafür, welche Ziele in welcher Tiefe am Ende der jeweiligen Klassenstufe erreicht werden müssen bzw. vor Beginn des Informatikunterrichts in Klassenstufe 7 vorausgesetzt werden. In den nächsten Jahren werden sich diese Voraussetzungen auf Grund der parallelen Einführung der Lehrpläne erst schrittweise einstellen.
Vielleicht kann diese Tabelle die Basis für schulinterne Fortbildungen bilden, um zu charakterisieren, mit welchen Vorleistungen auch bei der Computernutzung im Fachunterricht zu rechnen ist und welche Fertigkeiten die Schüler dort vertiefen.
Keinesfalls sollten die Übersichten als Dogma verstanden werden, eher als Anregung zur weiteren Konkretisierung der angegebenen Ziele. Des Weiteren bilden diese Tabellen natürlich die Basis und eine gewisse Orientierung für die Fortsetzung der fachsystematischen informatischen Bildung in den Klassenstufen 9 und 10. Die integrierte Behandlung von Anteilen der informatischen Bildung in den Lehrplänen aller Fächer verlangt eine explizite Untersetzung der informatischen Zielebenen und deren Verwirklichung im jeweiligen Fach. Es muss gesichert werden, dass der Informatikunterricht sowohl Vorleistungen für die Nutzung von Informatiksystemen in anderen Fächern als auch die Anschlussfähigkeit der Abschlüsse der Oberschule zur beruflichen Bildung leistet.

3 Jahrgangsziele und ihre Umsetzung

Bei der Unterrichtsplanung, der Auswahl der Themen und Werkzeuge und der methodischen Gestaltung ist darauf zu achten, dass die Bedienung der Werkzeuge nicht im Vordergrund steht. Vielmehr ist eine Ausgewogenheit zwischen Erarbeitung von informatischen Grundlagen und deren praktischer Umsetzung in verschiedenen Systemen gefordert.
Die Wahl der geeigneten informatischen Anwendungen für die jeweiligen Ziele und Inhalte der Lernbereiche obliegt dem unterrichtenden Fachlehrer. Bei der eigenen Entscheidung über die Folge der Anwendungen sind die Festlegungen im Lehrplan zu beachten.
Für die Jahrgangsstufen 7 und 8 sind folgende Anwendungsbereiche vorgeschrieben:

- Informationen in Tabellen darstellen
- Tabellen zum Kalkulieren nutzen
- Informationen in grafischer Form präsentieren

Insbesondere sind bei der Wahl der Anwendung in Klassenstufe 8 folgende Aussagen zu beachten:

- Die Schüler lernen eine weitere Anwendung nutzen und erkennen deren Vorteile bei der Lösung einer neuen Aufgabenklasse.

Aus diesen Lehrplanfestlegungen ergeben sich z. B. folgende Varianten der Nutzung besonders geeigneter Anwendungen:

Klassenstufe	Variante 1	Variante 2	Variante 3
7	Tabellenkalkulation Präsentationsprogramm / Textverarbeitung	Pixelgrafik Tabellenkalkulation	Textverarbeitung Pixel / Vektorgrafik
8	Vektorgrafik Programmieren, z. B. mit Scratch	Textverarbeitung Programmieren, z. B. mit Karol the Robot	Präsentationsprogramm Programmieren mit Tabellenkalkulation und Makros / Programmieren mit Logo

Die dargestellten Linienführungen der Anwendungen basieren nicht zwingend auf der Abfolge der einzelnen Lernbereiche. Vielmehr ist es notwendig, den jeweiligen fachlichen Themen geeignete Anwendungen zuzuordnen und daran mehrere Zielebenen parallel umzusetzen.
Unter diesem Blickwinkel kann kein Katalog von Mindestforderungen bezüglich des Umgangs mit einer bestimmten Anwendung aufgestellt werden.

Bei der Entscheidung für eine vorgestellte oder eigene Variante ist eine Abstimmung in den Fachkonferenzen der Schule notwendige Voraussetzung. Dabei ist den Anforderungen aus Schulkonzepten, aus Lehrplänen anderer Fächer (Web: [4]) und fächerverbindender Arbeit (Web: [5]) Rechnung zu tragen. Die Abstimmung muss auch die technischen und schulorganisatorischen Voraussetzungen berücksichtigen.
Es zeigt sich:
Die geforderte Bedienung und Beherrschung von Anwendungen ist nicht alleiniger Gegenstand des Informatikunterrichts. Sie verlangt neue Formen und Wege der Abstimmung und Zusammenarbeit zwischen Schulleitung, den Kollegen anderer Fächer und den Informatiklehrern.

Planung eigener Varianten:

Klassenstufe	Variante 4	Variante 5
7		
8		

4 Darstellung der Varianten – ein möglicher Gang durch den Unterricht

Die Darstellung möglicher Varianten soll die Flexibilität in der Gestaltung des Unterrichts befördern. Sie soll helfen bei der Planung langfristig zu denken und Lernbereiche nicht unbedingt in der Folge des Lehrplanes zu bearbeiten. Es wird zum Verständnis dringend empfohlen, die vorangehenden Abschnitte zur Erläuterung der Jahrgangsziele und zur Auswahl möglicher Applikationen zu beachten. Dadurch sollte klar werden, dass die im Folgenden dargestellten Varianten Möglichkeiten darstellen, wie Unterricht ablaufen könnte. Es wurde insbesondere darauf Wert gelegt, die Inhalte im Zusammenhang mit fachlichen und methodischen Bemerkungen zu beschreiben und eine Zuordnung zu entsprechenden Jahrgangszielen und Lernbereichen vorzunehmen. Auf eine Angabe des zeitlichen Umfangs wurde bewusst verzichtet.
Die Übersichten sollen Anregungen geben, eine eigene Variante zu entwickeln sowie sie als Basis für die Unterrichtsplanung und die eigene Stoffverteilung zu nutzen.

Variante 1:

Klassenstufe 7

Mit Objekten arbeiten

Inhalt	**Bemerkung zur Umsetzung**	LB	JSZ
Wer war der Beste beim Sportfest?	Einführung der Tabellenkalkulation über reale Problemstellung Fachübergreifendes Arbeiten, z. B. • Auswertung der Daten eines Sportfestes • Temperatur-Zeit-Diagramm aus Physik Klasse 6 Schüler benutzen das vorbereitete Rechenblatt und erforschen dessen Funktionalität	2	B
Rechnen mit Zellen, Zeilen und Spalten	Einführung der Datenstruktur in der Tabellenkalkulation Struktur des Einführungsbeispiel untersuchen Einführung der Objekte: • Zelle E4 – allgemein Zelle (Spaltenbuchstabe/Zeilenzahl) • Spalte A – allgemein … • Zeile 4 – allgemein … • Rechenblatt Sportfest – allgemein … Einführung der Begriffe Objekt, Attribut und Attributwert am einfachen Beispiel aus der Erfahrungswelt der Schüler (nichtinformatisch) pauls_t_shirt material = Baumwolle farbe = weiß groesse = XXL … …	2	A, C
	Definition der Begriffe Objekt, Attribut und Attributwert und allgemeine Darstellung (siehe Kapitel 5/6) objeket_name attribut1= Attributwert attribut2= Attributwert attribut3= Attributwert … … Übertragen der Begriffe auf das Einführungsbeispiel		

Hinweis für den Lehrer: In Hinblick auf die Klassendefinition in der Klassenstufe 8 muss immer mit konkreten Objekten gearbeitet werden. Klasse „ZELLE“, Objekt „zelle_e4“

Hinweis für den Lehrer: Bezogen auf die Algorithmierung sollte die Objektdarstellung von Anfang an nur mit Kleinschreibung und ohne Sonderzeichen (außer dem Unterstrich) verwendet werden.

<table>
<tr><th>Inhalt</th><th>Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Aus Rot
mach Grün</td><td>Operation: Ändern von Attributwerten
Einführung des Begriffs Operation am T-Shirt-Beispiel:
<table>
<tr><th colspan="2">pauls_t_shirt</th></tr>
<tr><td>material=</td><td>Baumwolle</td></tr>
<tr><td>farbe=</td><td>rot</td></tr>
<tr><td>groesse=</td><td>XXL</td></tr>
</table>
faerben→
<table>
<tr><th colspan="2">pauls_t_shirt</th></tr>
<tr><td>material=</td><td>Baumwolle</td></tr>
<tr><td>farbe=</td><td>grün</td></tr>
<tr><td>groesse=</td><td>XXL</td></tr>
</table>

Definition der Begriffe Operation und allgemeine Darstellung
<table>
<tr><th colspan="2">objekt_name</th></tr>
<tr><td>attribut1=</td><td>Aw</td></tr>
<tr><td>attribut2=</td><td>Aw</td></tr>
<tr><td>attribut3=</td><td>Aw</td></tr>
</table>
operation →
<table>
<tr><th colspan="2">objekt_name</th></tr>
<tr><td>attribut=</td><td>Aw</td></tr>
<tr><td>attribut=</td><td>geänderter Aw</td></tr>
<tr><td>attribut=</td><td>Aw</td></tr>
</table>
</td><td>2</td><td>A, C</td></tr>
<tr><td>Tabellen helfen in vielen Bereichen</td><td>Lösen einfacher Aufgabenstellungen mit der Tabellenkalkulation
Zielstellung ist die Beherrschung der Arbeit mit der Tabellenkalkulation bis zum Erstellen von Diagrammen.
• Flaschenpfandberechnung
• Vergleich von Anbietern (Handy, Auto, Energieanbieter, ...)
• Notenspiegel</td><td>2</td><td>B</td></tr>
<tr><td>Immer eine aktuelle Notenübersicht</td><td>Lösen einer einfachen Aufgabenstellung mit einer anderen Kalkulationssoftware
Zielstellung ist die Übertragung des Modells Objekt-Attribut-Attributwert auf ein anderes Programm
• z. B. noch mal Notenspiegel
Erkennen der Unterschiede bei Anzahl der Attribute und den zugehörigen Werten.</td><td>2</td><td>A</td></tr>
</table>

Daten und Informationen – Verschlüsselungen und Codes

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Wenn Cäsar zum Verschlüsseln schon einen Computer gehabt hätte	Anwenden der Problemlösung mit der Tabellenkalkulation im Wahlpflichtbereich 2 Was ist ein Code? – Beispiele: ASCII, Hieroglyphen, . . . Was ist eine Verschlüsselung? – Beispiel: Cäsar-Verschlüsselung ohne und mit Schlüsselwort Planung, Umsetzung, Kontrolle und Wertung einer kleinen Problemstellung mit der Tabellenkalkulation • Cäsar-Verschlüsselung ohne Schlüsselwort (Web: [6])	WP2	C, D

Aufbau und Funktion des Computers

<table>
<tr><th>Inhalt</th><th>Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Von Cäsar zu Enigma – Die Geschichte der Rechentechnik</td><td>Erstellung einer Tabelle zur Entwicklung der Rechentechnik
Nutzung des Internets zur Recherche
Achten auf exakte Quellenangabe in der Tabelle</td><td>WP2
LB1</td><td>B</td></tr>
<tr><td>Was braucht eine Rechenmaschine um rechnen zu können?</td><td>Eingabedaten – Rechenwerk – Ergebnisdaten
Wiederholung des EVA-Prinzips aus TC Klasse 5 (Hinweis: Prüfung der Vorkenntnisse)
Übertagen des EVA-Prinzips auf Vorgänge aus dem Alltag</td><td>1</td><td>B</td></tr>
<tr><td>Von EVA zum Blockschaltbild
(offene Unterrichtsform)</td><td>
<table>
<tr><th>Nutzung der Präsentationssoftware</th><th>Inhalte des LB</th></tr>
<tr><td>Einführung der Anwendung Präsentation als Grundlage zum Erstellen des Blockschaltbildes</td><td></td></tr>
<tr><td>Modell Objekt-Attribut-Attributwert in einem Präsentationsprogramm</td><td>einfaches Blockschaltbild zu EVA aus TC</td></tr>
<tr><td></td><td>Begriffe: Hardware, Prozessor, Bus, Speicher</td></tr>
<tr><td>Erstellung von einzelnen Objekten der Präsentation (z.B. Rahmen für externe Speicher, Zentraleinheit, BUS, … Blockpfeile für den Informationsfluss) für die Erweiterung des Blockschaltbildes (Gruppenarbeit)</td><td>Übersicht zu Arten externer Speicher
Aufbau der Zentraleinheit (Arbeitsspeicher, Prozessor mit Rechenwerk und Steuerwerk)
Bussystem als Träger des Informationsflusses</td></tr>
<tr><td>Einfügen der neuen Module in das Blockschaltbild EVA</td><td></td></tr>
</table>
</td><td>1</td><td>A, C</td></tr>
<tr><td>Speichern des Blockschaltbildes zum Mitnehmen</td><td>Was sagt uns die Größe der Datei?
Maßeinheiten für Speichergrößen
Speicherkapazität verschiedener externer und interner Speichermedien
Was wird eigentlich gespeichert?
Unterschied Daten, Informationen und Datei
Interne Darstellung von Daten als binäre Zustände (methodischer Hinweis: Öffnen der Blockschaltbilddatei in der hexadezimalen Ansicht eines Editors)</td><td>1</td><td>A</td></tr>
</table>

Strukturieren von Daten

<table>
<tr><th>Inhalt</th><th colspan="2">Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Wie strukturiere ich sinnvoll meine verschiedenen Daten aus dem Schuljahr (Daten der Tabellenkalkulation und Grafikdaten)?</td><td colspan="2">Wiederholung der Begriffe: Daten, Datei
Einführung der Begriffe Dateityp und Verzeichnis
Einführung der Verzeichnisstruktur unter Nutzung von Analogien aus anderen Fächern</td><td>1</td><td>A, B</td></tr>
<tr><td>Meine Verzeichnisstruktur</td><td>Nutzung der Präsentationssoftware</td><td>Inhalte des LB</td><td>1</td><td>A</td></tr>
<tr><td></td><td></td><td>Planung einer sinnvollen eigenen Verzeichnisstruktur
Netzwerk</td><td></td><td></td></tr>
<tr><td></td><td>Visualisierung der geplanten Struktur mittels Objekte der Präsentation</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Begriffe: Software, Programm
Beispiele für Programme zur Erstellung von Verzeichnisstrukturen</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Umsetzen der geplanten Verzeichnisstruktur</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Speichern von Daten in eigener Verzeichnisstruktur</td><td></td><td></td></tr>
</table>

Klassenstufe 8

Vom Objekt zur Klasse

<table>
<tr><th>Inhalt</th><th>Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Wir planen die Gestaltung unseres Klassenzimmers – es gibt verschiedene Varianten – wie kann man diese effektiv darstellen?</td><td>
Einführung der Anwendung Vektorgrafik als Werkzeug
Es gibt mehrere verschiedene Objekte mit gleichen Eigenschaften (Stühle, Schülertische, …).
Finden von Eigenschaften des Schülertisches im Klassenzimmer
Strukturieren der Eigenschaften

Einführung des Klassenbegriffs am Beispiel Schülertisch
<table>
<tr><td colspan="2">peters_tisch: TISCH</td></tr>
<tr><td>farbe=</td><td>beige</td></tr>
<tr><td>hoehe=</td><td>800 mm</td></tr>
<tr><td>breite=</td><td>800 mm</td></tr>
<tr><td>laenge=</td><td>1200 mm</td></tr>
<tr><td>position=</td><td>2500 mm, 7500 mm</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td colspan="2">tisch_farbe_aendern()
tisch_erzeugen()
tisch_position_aendern(x,y)
…</td></tr>
</table>
Übertragen der Klassendefinition auf Beispielklasse Schülertisch
<table>
<tr><td colspan="2">TISCH</td></tr>
<tr><td>farbe:</td><td>weiß, blau, rot, beige, …</td></tr>
<tr><td>hoehe:</td><td>800 mm … 1000 mm</td></tr>
<tr><td>breite:</td><td>800 mm … 1200 mm</td></tr>
<tr><td>laenge:</td><td>800 mm … 1800 mm</td></tr>
<tr><td>position:</td><td>0 … 10000 mm, 0 … 16000 mm</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td colspan="2">tisch_farbe_aendern ()
tisch_erzeugen ()
tisch_position_aendern (x,y)
…</td></tr>
</table>
Übertragen des Klassenmodells auf Beispiele aus der Erfahrungswelt (Autos, Pflanzen, Berge, …) – Definition der Begriffe Klasse, Attributwertebereich, Methode und Erweiterung der allgemeinen Darstellung (siehe Kapitel 5/6)
<table>
<tr><td colspan="2">NAME_DER_KLASSE</td></tr>
<tr><td>attribut1:</td><td>attributwertebereich</td></tr>
<tr><td>attribut2:</td><td>attributwertebereich</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td colspan="2">methode1 ()
methode2 ()
…</td></tr>
</table>
</td><td>1</td><td>A, B, C</td></tr>
</table>

<table>
<tr><th>Inhalt</th><th>Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Welche Klassen brauchen wir zur Darstellung des Klassenzimmers?</td><td>Klassen in der Anwendung Vektorgrafik (Umfang der Attribute und Methoden abhängig vom verwendeten Vektorgrafikprogramm)
<table><tr><td colspan="2">LINIE</td></tr><tr><td>linien_farbe:
linien_straerke:
linien_art:
…</td><td>weiß, blau, rot, schwarz, …
Haarlinie; 0,5 pt; …
durchgängig, gepunktet, …
…</td></tr><tr><td colspan="2">linie_erzeugen()
linien_farbe_aendern()
…</td></tr></table>
<table><tr><td colspan="2">FLAECHE</td></tr><tr><td>flaechen_linien_farbe:
flaechen_linien_straerke:
flaechen_fuell_farbe:
…</td><td>weiß, blau, rot, schwarz, …
Haarlinie; 0,5 pt; …
durchgängig, gepunktet, …
…</td></tr><tr><td colspan="2">flaeche_erzeugen()
flaechen_linien_farbe_aendern()
flaechen_fuell_farbe_aendern()
…</td></tr></table></td><td>1</td><td>A, C</td></tr>
<tr><td>Wir planen Varianten unser Klassenzimmer mit der Anwendung Vektorgrafik</td><td>Erarbeiten einer Vorlage mit:
• Raumgrundriss
• ein Objekt Schülertisch als Vorlage
• ein Objekt Schülerstuhl als Vorlage
• weitere Objekte des Zimmers
Jeder Schüler erstellt aus den Vorlagen eine eigene Variante der Zimmereinrichtung.</td><td>1</td><td>B, D</td></tr>
</table>

Vom Algorithmus zum Programm

<table>
<tr><th>Inhalt</th><th>Bemerkung zur Umsetzung</th><th>LB</th><th>JSZ</th></tr>
<tr><td>Wie bringen wir den Computer dazu, etwas auf unsere Anweisung „selbst zu tun“? - Warum lernen wir programmieren?</td><td>Beispiele für das Programmieren im Alltag bei Videorecorder, Handy, Waschmaschine, …
Beschreiben von Handlungsabläufen wie Abwaschen, Staubsaugen, …
Einführung des Begriffs Algorithmus, Übertragen des Begriffs auf Probleme und Handlungsabläufe im Alltag mit Verdeutlichen der Algorithmus-Eigenschaften</td><td>2</td><td>B, C</td></tr>
<tr><td>Eine Katze lernt Anweisungen zu befolgen …</td><td>Einführung in die Welt von Scratch
Klassen (Bühne, Katze, Ball)
Methoden, die Katze tätig werden lassen (‚Bewegung’ & ‚Aussehen’ & ‚Malstift’):
gehe_n-er-Schritt, sage:“xyz“ , drehe_dich_um_n_Grad,
hinterlasse_Abdruck

Methoden, die Katze etwas melden (‚Fühlen’):
wird_Rand_berührt?, wird_Objekt_berührt?, Taste_Leertaste_gedrückt?

Programmstruktur: Folge am Beispiel Scratch und allgemein
Katze steht am Strand und will fünf Überschlagsprünge machen.

Darstellung der Folge
start
gehe_n-er-Schritt
drehe_dich_um_n_Grad links
gehe_n-er-Schritt
drehe_dich_um_n_Grad rechts
stop

Anweisung1
Anweisung2
Anweisung3
Anweisung4

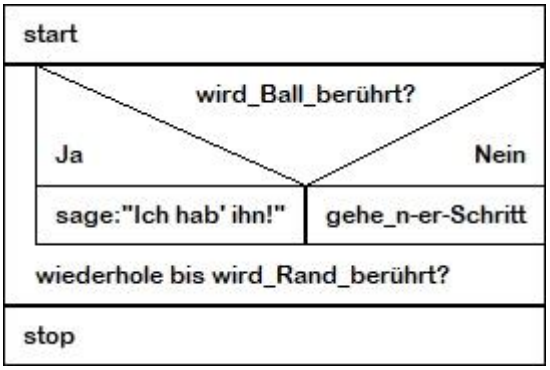
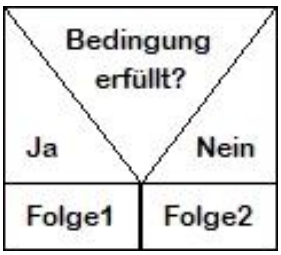
Programmstruktur: Wiederholung am Beispiel Scratch und allgemein:
Gehe drei n-er Schritte.

gehe_n-er-Schritt
Wiederhole bis drei erreicht

gezählte Wiederholung
Folge
Wiederhole bis Anzahl erreicht

Wiederholung mit vorangestelltem Test
Wiederhole solange Bedingung wahr
Folge

Wiederholung mit nachgestelltem Test
Folge
Wiederhole bis Bedingung wahr</td><td>2</td><td>A, B, C</td></tr>
</table>

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Eine Katze entscheidet sich	Programmstruktur: Verzweigung am Beispiel Scratch und allgemein: Wenn Katze unterwegs auf den Ball trifft, soll sie sagen:“Ich hab’ ihn!“.   	2	A, B, C
Probleme, Probleme, Probleme - Wie lösen wir sie?	Schritte des Problemlöseprozesses (Problemanalyse, Modellbildung, Implementierung, Test) mit Dokumentation. Übertragen auf ein Beispielproblem aus Scratch Katze soll immer gerade aus laufen. Trifft er auf den Ball soll er sagen:“Ich hab’ ihn!“ **Problemanalyse:** Welche möglichen Situationen können in der geänderten Welt auftreten? **Lösungsentwurf:** Aufstellen eines Struktogramms **Umsetzung:** Implementierung des Struktogramms in Scratch **Test:** In verschiedenen Welten – Kritik und Korrektur Dokumentation der Problemstellung, Lösung unter bestimmten Bedingungen, Grenzen der Lösung, Handhabung des Programms Selbständiges Lösen eines neuen Problems nach der Schrittfolge des Problemlösungsprozesses in offener Unterrichtsform (Partnerarbeit)	2	C, D

Codieren

Inhalt	**Bemerkung zur Umsetzung**	LB	JSZ
Katze lernt Morsen	Beispiele für maschinelle Codierungen (Morse-Code, ASCII-Code, …) Problem - Morsen mit Katze Problemanalyse: • Wie stellt Katze ein Morsezeichen dar? • Methode hinterlasse_Abdruck für Punkt • Methode hinterlasse_Abdruck → Methode gehe_10er-Schritt → Methode hinterlasse_Abdruck • für Strich • Methode gehe_100-er-Schritt für Trennzeichen . **Lösungsentwurf:** • Aufstellen der einzelnen Darstellungsformen für jeden Buchstaben: Wort C A T • Aufstellen einer Programmstruktur als Folge **Umsetzung – Test – Dokumentation:** • Partnerarbeit Codieren, Decodieren • Schüler „lesen“ gegenseitig ihre Scratch-Morsezeichen • Grenzen der Umsetzung mit Scratch • Erstellen der Dokumentation	WP2	C, D

Variante 2

Klassenstufe 7

Es wurde in Klasse 7 der Wahlpflichtbereich "Computer Gestern – Heute – Morgen" gewählt. Er wurde in die Pflichtbereiche integriert (siehe Spalte Lernbereiche LB).

Vorleistung aus TC Klasse 5 und 6 beachten

Computer verstehen unter Anwendung der Pixelgrafik

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Fachbegriffe zum prinzipiellen Aufbau des Computers und der verwendeten Software	Analyse von aktuellen Computerangeboten (Anzeigen, Prospekte, Onlineangebote) Auf der Basis von Werbeprospekten werden Gemeinsamkeiten bezüglich der Hardware (Prozessor, Speicher, Festplatte …) und der Software (Systemsoftware, Anwendersoftware) aller Angebote herausgearbeitet.	1, WP3	B, C
Blockschaltbild zum Aufbau eines Rechners	Nutzung eines Programms zur Erstellung von Vektorgrafiken (z. B. OpenOfficeDraw, Microsoft Draw, …)	1, WP3	C
Anwenden des EVA - Prinzips auf Alltagsvorgänge	Vorstellung der Handhabung technischer Geräte, z. B. Parkschein-, Getränkeautomat Übung der Darstellung als Pixelgrafik	1, WP3	C
Notwendigkeit einer geordneten Datenaufbewahrung und Gestalten von Verzeichnisstrukturen	Bekannte Einordnungsverfahren anderer Fächer (z. B. Geografie: Länder-Kontinente, Biologie: Gattungen und Arten, …) dienen als Ausgangspunkt für die Einführung der Baumstruktur (handschriftlich oder Mind Map) Nutzung eines Dateimanagers zur Umsetzung in einem Verzeichnisbaum Einführung der Begriffe: Programm, Daten, Datei, Verzeichnis, Dateityp, Dateiname Die Schüler strukturieren ihre bisherige Dateisammlungen (auch aus allen anderen Fächern).	1	A, D
Einblick in die Geschichte der Rechentechnik.	Gestalten eines Zeitstrahles zu den Etappen der Entwicklung der Rechentechnik mit Hilfe der Pixelgrafik	1, WP3	B

Objekte – Attribute – Operationen in der Tabellenkalkulation

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Historische Entwicklung des Rechnens, Vorteile der Tabellenkalkulation	Vergleich: schriftliches Rechnen → Taschenrechner → Tabellenkalkulation Wie oft hast du schon Zähne geputzt? Wie alt bist du? (Tage, Stunden, ...)	2	B, C
Grundfunktionen bei der Bedienung eines Rechenblattes	Am Beispiel Kostenplanung für eine Klassenfahrt Begriffe: Zeile, Spalte, Zelle, Adresse	2	A, C
Das Modell Objekt-Attribut-Attributwert	Übertragen des Modells auf Beispiele außerhalb der Tabellenkalkulation (Spielkarten, ...) golf_gti_von_opa leistung = 115 PS farbe = rot preis = 20.000 € ... mein_fahrrad ganganzahl = 27 farbe = rot preis = 800 € ...	2	A, C

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Das Objekt Zelle E5, seine Attribute und Attributwerte	Arbeitsblatt mit einfacher Darstellung der Objekte am Beispiel Kostenplanung für eine Klassenfahrt zelle_e5 inhalt = Kosten textdarstellung = Verdana 12pt rechtsbündig hintergrund = weiß rahmen = einfach schwarz Umriss ...	2	A, C
Zahlenformate	Arbeitsblatt zu Zahlenformaten Begriff „Operation“ einführen Unterschied zwischen Operation (z. B. Formatieren) und Aktion (z. B. Ausschneiden, Einfügen) erklären	2	A, C
Anwendungsaufgabe: Rechnung für einen Hardwareeinkauf	Benutzung eines anderen Kalkulationprogramms Zuordnung von Objekten, Attributen, ... Herstellung eines Bezugs zum LB1	2	A, B, C, D
Diagramme (Erstellung, Arten)	Beispiel „Zensurenverteilung der letzten Leistungskontrolle“ Freie Programmwahl Zuordnung von konkreten Objekten zum Modell Einfache Darstellung diagramm_zensurenverteilung art = Säule daten = B3:G4 x_achse = Zensur y_achse = Anzahl titel = Zensurenverteilung ort = eigenes Blatt name = Diagramm_Zensurenverteilung	2	A, C
Komplexe Übungen mit einfachen Rechenaufgaben mit Diagramm	z. B. Bodymaßindex z. B. Durchschnittsverbrauch eines Mopeds Wechsel des Programms	2	A, C, D

Ein zweites Programm der Anwendung Tabellenkalkulation ist erforderlich.

Klassenstufe 8

Klassen und Objekte am Beispiel der Textverarbeitung

Die Stunden-mitschriften werden im gesamten Lernbereich 1 in Form von Textdokumenten erstellt.

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Erstellen eines Textes	Anknüpfen an Kenntnisse zur Textverarbeitung aus Technik und Computer Beherrschung von Grundregeln der Texteingabe und Textformatierung	1	A, B, C
Übertragen des Objekt-Attribut-Attributwert-Modells auf die Textverarbeitung	am Beispiel des erstellten Textes **zeichen_e** schriftart = Arial schriftgroesse = 12 Punkt schriftstil = kursiv ... **absatz_3** ausrichtung = Blocksatz einzug_links = 4 cm zeilenabstand = doppelt ...	1	A, C
Klassen in der Textverarbeitung • ZEICHEN • ABSATZ • DOKUMENT	Finden von Gemeinsamkeiten verschiedener Objekte eines Beispieltextes Vergleichen mehrerer Absätze: Überschrift, 3. Absatz, ... , 8. Absatz Finden gleicher Attribute Verallgemeinern: gehören zur gleichen Klasse ABSATZ Definition des Klassenbegriffes	1	A, C
Darstellung der Klassen • ZEICHEN • ABSATZ • DOKUMENT	Erstellen der Klassendarstellung mit einer Textverarbeitung **ZEICHEN** schriftgroesse: 2pt ... 128pt schriftfarbe: rot, blau, grün, ... schriftstil: kursiv, fett, ... schriftfarbe_aendern () ... **ABSATZ** ausrichtung: links, rechts, Blocksatz, zentriert zeilenabstand: einfach, ..., mehrfach tabulator: 0,1 cm links ohne Füllzeichen, 2,5 cm zentriert Füllzeichen < >, 3,9 cm dezimal Füllzeichen <--->, ... ausrichtung_aendern () ...	1	A, C

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Methoden der Klasse „ABSATZ“	Definition des Begriffs „Methode“ Methoden: Tabulator setzen, Einzug einrichten, Ausrichtung zuweisen, …	1	A, C
Übertragen des Modells „Klasse“ auf Erfahrungen aus dem Alltag und der Informatik	SCHUELER name: Meier, Schulze, … alter: 7 bis 18 Jahre geschlecht: m, w … loesche_schueler() ... ZELLE inhalt: 13, =B4+B5, Zensurenübersicht zahlendarstellung: Währung in €, Zahl mit zwei Dezimalstellen, Prozentwert, … textdarstellung: Verdana 12pt Standard schwarz rechtsbündig, ... hintergrund: gelb, ... rahmen: Linie einfach einfach schwarz Umriss, … … nach_rechts_ausfuellen() ...	1	A, C

Klassen und Objekte in Computerspielen

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Computerspiele	Analysieren der Elemente eines Spieles Beispiel: Autorennen RENNAUTO fahrzeugtyp: Ferrari, …, BMW leistung: 300 PS, …, 900 PS farbe: rot, …, silber … beschleunigen() reifen_wechseln() ... schumis_auto: RENNAUTO fahrzeugtyp= Ferrari leistung= 850 PS farbe= rot … beschleunigen() reifen_wechseln() ...	3	C

Algorithmenstrukturen mit dem Roboter Karol

Inhalt	Bemerkung zur Umsetzung	LB	JSZ
Begriff „Algorithmus“ und dessen Merkmale	Beispiele aus der Erfahrungswelt des Schülers: • Backen eines Kuchens • Erstellen eines Textes	2	C

Programmstrukturen in alltäglichen Abläufen und deren Darstellung	**Folge:** Vom Wecker bis zur Schule
	Wecker ruhig stellen
	Aufstehen
	Waschen
	Anziehen
	Frühstück
	…
	Betreten der Schule
	
	Wiederholung: 10 Kartoffeln schälen
	Wiederhole 10 mal
	  Kartoffel aufnehmen
	  Kartoffel schälen
	  Kartoffel ablegen
	
	Verzweigung: Essenausgabe in der Schulküche
	Möchtest du das Essen A?
	ja / nein
	Essen A | Essen B
		2	C

Anwendung der Programmstrukturen im Programm Karol (Web: [8])	Folge:
	• Ablaufen einer Strecke
	• `(Schritt, Schritt, Schritt, Schritt, Schritt)`
	• Aufnahme, Transport und Ablegen eines Steines
	• `(Aufheben, Schritt, Hinlegen)`
	Wiederholung:
	• Ablaufen einer Strecke
	`(Wiederhole 5 mal Schritt)`
	• Laufen bis zur nächsten Wand
	`(Solange NichtIstWand, Schritt)`
	Verzweigung:
	• Drehen vor einer Wand
	`(wenn IstWand dann LinksDrehen sonst Schritt)`
	• Aufheben eines Steines
	`(wenn IstStein dann Aufheben sonst Schritt)`
		2	C

Um Verständnis für die Arbeitsweise von Karol zu erlangen sollte zu Beginn die Bedienung im Direktmodus benutzt werden.

Inhalt	Bemerkung zur Umsetzung	LB	JSZ

Darstellen der Programmstrukturen in Karol als Struktogramm
Interpretation vorgegebener Struktogramme
Umsetzen von Struktogrammen in Programmausführungen
	Kombination der einzelnen Programmstrukturen an einfachen Beispielen
	
	z. B. Karol baut eine Mauer um seine Welt mit einer Höhe von 3 Steinen
	
	`wiederhole 12 mal`
	`  solange NichtIstWand tue`
	`  Hinlegen`
	`  Schritt`
	`  *solange`
	`  LinksDrehen`
	`*wiederhole`
	
	wiederhole 12 **mal**
	solange NichtIstWand
	Hinlegen
	Schritt
	LinksDrehen
	
	z. B. Was macht Karol im folgenden Beispiel?
	
	wiederhole 8 **mal**
	wiederhole 5 **mal**
	Hinlegen
	Schritt
	LinksDrehen
	wiederhole 3 **mal**
	Hinlegen
	Schritt
	LinksDrehen
	
	(Karol baut ein Schwimmbad 6 Steine lang, 4 Steine breit und 4 Reihen hoch.)
	
	Umsetzen des Struktogramms in ein ausführbares Programm
		2	A, C

Lösen von einfachen Problemen mit Karol Lösungsentwurf → Struktogramm Umsetzung → Programmcode Test → Bewerten des Programmablaufs	Beispiel 1: Treppe Beispiel 2: Burg (Beispiele im Anhang an Tabelle)	2	C, D

Anmerkung: In der Klassenstufe 7 wurde als dritter Lernbereich der Wahlpflichtbereich “Computer Gestern – Heute – Morgen“ gewählt. Er wurde in die Pflichtbereiche integriert (siehe Spalte Lernbereiche LB).

Anhang:

Programmbeispiele zum Lösen einfacher Probleme mit der Programmierumgebung Karol:

Beispiel1: Karol baut eine Treppe.
(Welt: 10*10*10)


```
wiederhole 9 mal
      MarkeSetzen
      solange NichtIstWand tue
           Hinlegen
           Schritt
      *solange
       LinksDrehen
       LinksDrehen
       solange NichtIstMarke tue
            Schritt
       *solange
       LinksDrehen
       LinksDrehen
       Schritt
*wiederhole
```

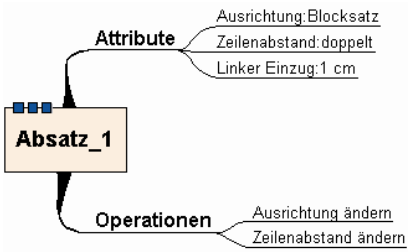
Beispiel 2: Karol baut eine Burg.
(Größe der Welt, Mauerhöhe 4, ungerade Kantenlänge!)


```
wiederhole 16 mal
       solange NichtIstWand tue
             Hinlegen
             Schritt
       *solange
        LinksDrehen
*wiederhole
 wiederhole 4 mal
       solange NichtIstWand tue
            Hinlegen
            Schritt
            Schritt
       *solange
        LinksDrehen
 *wiederhole
```

Variante 3:

Klassenstufe 7

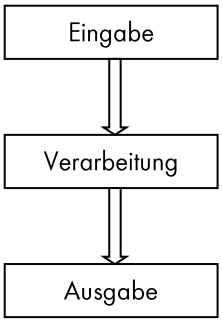
Objekt-Attribut-Attributwert mittels Textverarbeitung

Inhalte	Bemerkung zur Umsetzung	LB	JZ
Kennen grundlegender Datenstrukturen der Textverarbeitung: Objekt, Attribut, Attributwert, Operation Zuordnung: Objekt - Attribut	Dokument (Seitenformat, Querformat, Speichern), Absatz (Ausrichtung, zentriert, Verschieben) Zeichen (Farbe, rot, Farbe ändern) Inhalte aus Erfahrungsbereich der Schüler, fächerübergreifend (z. B. Deutsch – Gedicht)	2	A, C
Darstellungsform für Datenstrukturen	Mind Map (Web: [9])  a schriftart= Arial schriftstil= fett schriftgroesse= 16 pt ... schriftart_aendern() ...	2	C
Lösen von typischen Aufgaben der Textverarbeitung	Brief, Gedicht, Stichwortzettel, Einladung	2	D
Übertragung der Kenntnisse auf anderes Programm derselben Anwendung	z. B. OpenOffice Writer, Libre Office Writer, Microsoft Word Objekt-Attribut-Attributwert-Beziehungen untersuchen Allgemeingültigkeit und Notwendigkeit des Objekt-Attribut-Attributwert-Modells herausarbeiten	2	B, C
Lizenzrecht / Urheberrechte	Nutzung und Beschaffung von Software (z. B. Programme, Musik, Grafik)	2	B

Verzeichnisstrukturen

Inhalte	**Bemerkung zur Umsetzung**	**LB**	**JZ**
Verzeichnisstrukturen auf Datenträgern Datei, Verzeichnis, Dateityp	Notwendigkeit demonstrieren Arbeitsplatz (Ordneroptionen), Dateimanager Untersuchen, planen, speichern auf Festplatte fächerübergreifende Beispiele aus Biologie, Geografie, TC)	1	A, C
Pfadangaben	Netzwerk, Web-Adressen	1	A
Software - Programme - Dateien -Daten	Zusammenhang / Hierarchien sichtbar machen	1	A

Aufbau und Arbeitsweise eines Rechners

Inhalte	**Bemerkung zur Umsetzung**	**LB**	**JZ**
Aufbau des Computers: Hardware, Prozessor, Bus, Speicher	Vorkenntnisse aus TC Exakte Definitionen Demontage eines Gerätes	1	C
Vergleich von Speichermedien	Bzgl. Flüchtigkeit, Kapazität (Maßeinheiten), Zugriffsverfahren, Speicherverfahren, Mobilität	1	C
Ein- und Ausgabegeräte	Analyse eines Rechnerarbeitsplatzes (TC)	1	
EVA-Prinzip	Funktionsprinzip von Informatiksystemen Sonderrolle der Speichermedien	1	C
Blockschaltbild zum Aufbau	 Informationsfluss darstellen, Grafikobjekte der Textverarbeitung als Werkzeug nutzen	1	C, D
Übertragung des EVA-Prinzips auf den Alltag	Demonstration an Geräten: Untersuchung, welche Bauteile welche Funktion haben; Zuordnung zum Blockschaltbild	1	B
Historischer Abriss	Nutzung von Internetangeboten	1	B

Wahlpflicht1: Computerspiele

Inhalt	**Bemerkung zur Umsetzung**	LB	JSZ
Sinn von Computerspielen	Vergleich mit herkömmlichen Spielen (Web: [10])	WP 1	B
Spielarten	Netzwerkspiele Strategie, Jump&Run, Abenteuer, Simulation, Logik Lernspiel Online	WP 1	
Benutzeroberflächen, Eingabegeräte	Gemeinsamkeiten und Unterschiede am Beispiel	WP 1	A

Achtung:
Bei Installation von Spielen Lizenzrechte beachten!

Klassenstufe 8

Wahlpflicht1: Computerspiele

Inhalte	**Bemerkung zur Umsetzung**	**LB**	**JZ**
Objekte, Attribute, Attributwerte in Simulationsspielen	z. B. Autorennen: Auswahl der Strecke, der Eigenschaften des Autos, des Fahrernamens auto_1: RENNAUTO farbe= rot marke= Ferrari leistung= 580 PS lenken() ...	WP 1	C
Untersuchung von Hardwarevoraussetzungen	Wiederholung zum Aufbau des Computers: Grafik, Sound, Eingabegeräte	WP 1	B, D

Klasse – Objekt – Attribut – Methode mittels Präsentation

Inhalt	**Bemerkung zur Umsetzung**	LB	JSZ
Klasse, Attribute, Attributwertebereich, Methoden	Einstieg über Klasse RENNAUTO möglich, Wertevorrat (-bereich) für Objekte wird in Klasse festgelegt, Methoden sind Operationen und Aktionen	1	C
Darstellung der Klassen	RENNAUTO farbe: rot, blau, weiß, . . . marke: Ferrari, Porsche, . . . leistung: 100 PS .. 700 PS lenken() ...	1	C
weitere Beispiele aus der Erfahrungswelt	geometrische Figuren, Schulklassen, Biologie	1	A, C
Einblick in Grundfunktionen, Einsatzgebiete einer Präsentationssoftware	Analyse einer vorhandenen, geeigneten Präsentationssoftware, z. B. PowerPoint, Impress (Star-Office)	1	B
Kennen grundlegender Datenstrukturen in Präsentationsprogrammen: Objekt, Attribut, Attributwert, Operation; Klasse, Attribut, Attributwertebereich, Methode	Klassen einer Präsentation: Dokument, Folie, Textfeld, Grafik, Link, Multimediaelemente FOLIE hintergrundfarbe: rot, blau, ... format: Bildschirm, A4, ... foliennummer: 1, ... duplizieren() ...	1	A, B
Erstellen einer Präsentation	z. B. Aufbau eines Rechners Aufgabe für Schüler: Ordne die verwendeten Objekte den Klassen der Präsentation zu und dokumentiere die Attributwerte!	1	A, C

Informationsverarbeitung Modell-Algorithmus-Lösung

Inhalte	Bemerkung zur Umsetzung	LB	JZ
Darstellung von Algorithmen	z. B. Ablaufbeschreibung, Folge verbaler Anweisungen, Befehlsfolge Problemstellung, die Grenzen der einfachen Befehlsfolge zeigt, z. B. im Getränkeautomat ist kein Wechselgeld vorhanden	2	C
Grundlegende Programmstrukturen: Folge, Wiederholung, Verzweigung Darstellung als Programmablaufplan	Telefon: • Folge, wenn optimaler Verlauf • Verzweigung, wenn besetzt (Wiederholung oder Abbruch) • Wiederholung, wenn besetzt • Abbruch, wenn Guthaben aufgebraucht Siehe Abbildung 1 im Anhang	2	C
Umsetzung der Strukturen mit Tabellenkalkulation	Folge: Berechne die Summe von zehn Zahlen! (Eingabe der Zahlen, Eingabe der Formel, Ausgabe des Ergebnisses) Siehe Abbildung 2 im Anhang Wiederholung: Erzeuge drei Zufallszahlen für ein Würfelspiel! (dreimalige Wiederholung der Eingabe der Funktion, Berechnung der Zahl, Ausgabe der Zahl) Siehe Abbildung 3 im Anhang Verzweigung: Berechne die Summe der drei Zufallszahlen. Prüfe auf Wert 18. Wenn ja, dann Ausgabe „Hauptgewinn“. Wenn nein, dann Ausgabe „Niete“! Siehe Abbildung 4 im Anhang	2	B, C

Inhalte	Bemerkung zur Umsetzung	LB	JZ
Problemanalyse, Lösungsentwurf, Umsetzung, Test, Dokumentation	Bearbeitung eines Beispiels Schritte einzeln behandeln z. B. fachübergreifend (Geo) Erzeugung eines Klimadiagramms aus den Monatswerten von Niederschlag und Temperatur	2	A, C
Lösen eines einfachen Problems	wichtig: selbstständige Schülerarbeit Beispiel: Lege eine Tabelle an, welche nach Eingabe der Einzelnoten den Durchschnitt berechnet und beim Überschreiten des Grenzwertes 3,5 eine Warnmeldung erzeugt!	2	A, D
Bewertung der Resultate	Überprüfung der Eigenschaften des Algorithmus Diskussion des Aufwand-Nutzen-Verhältnisses	2	A, B, D

Schlussbemerkungen:

Alle drei Varianten stellen unterschiedliche Herangehensweisen an die gesamte Planung der Klassenstufen 7 und 8 dar. Diese Varianten wurden von verschiedenen Arbeitsgruppen erstellt und sind deshalb in Darstellung und Formulierung verschieden.

Es wurde die Darstellung von Objekten und Klassen vereinheitlicht. Diese beruht auf einer didaktisch reduzierten und für die Oberschule angepassten Standarddefinition von UML (Web: [11]) und sollte in dieser Form verwendet werden (→ Erklärung Kapitel 5).

Bei der Erstellung eigener Varianten müssen die Klassenstufen 7 und 8 als Einheit betrachtet werden. Nicht zu vergessen ist, dass die Varianten der Klasse 7 und 8 richtungweisend auf die Herangehensweise an die Klassenstufen 9 und 10 wirken. Eine Vororientierung auf die Klassenstufen 9 und 10 erfolgt also bereits bei der Planung der Klassenstufen 7 und 8.

5 Begriffe und Aufgaben für den Unterricht

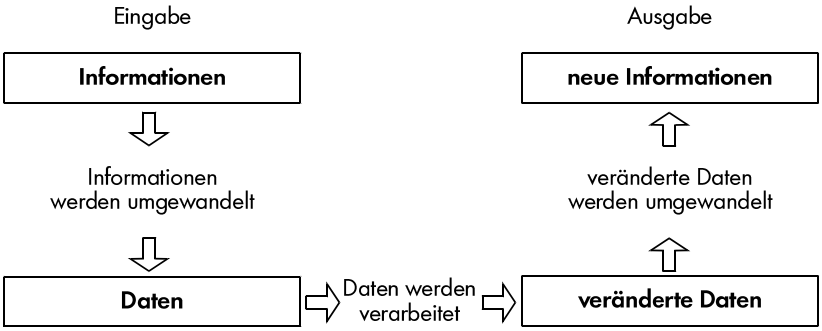
5.1 Ausgewählte Begriffe für den Schüler

Am Beispiel ausgewählter Begriffe, die im Unterricht der Klassen 7 und 8 eine Rolle spielen, soll verdeutlicht werden, welche Erwartungen an das Begriffswissen bei Schülern gestellt werden sollte. Dabei ist zu bedenken, dass dies stets unter dem Aspekt der didaktischen Vereinfachung geschieht und angegebene Erläuterungen im Unterricht als Definition bzw. Merksatz unmittelbar verwendet werden könnten.
In der Folge werden zuerst einige allgemeine Begriffe in alphabetischer Reihenfolge aufgeführt, deren Zuordnung auf eine Klassenstufe nicht möglich oder unnötig ist. Die anschließende Trennung der weiteren Begriffe nach den Klassenstufen 7 und 8 hat keinen Anspruch auf Vollständigkeit.
Außerdem sind weiterführende Erläuterungen zu ausgewählten Begriffen und zu grundlegenden Fachinhalten im Kapitel 6 zu finden.

Allgemeine Begriffe

Daten

Daten (Einzahl: Datum) sind Darstellungen von Informationen, die maschinell gespeichert, verarbeitet und transportiert werden. Sie werden als Folgen von Zeichen dargestellt. Im Computer werden diese Zeichen mittels 0 und 1, d. h. Strom fließt – Strom fließt nicht, codiert. (Print: [1])
Den Zusammenhang zwischen Information und Daten zeigt das folgende Bild:



Informatik

„Die Informatik beschäftigt sich als Wissenschaft mit der maschinellen Verarbeitung von Informationen mithilfe von Computern.“ (Print: [1])

Information

Eine Information ist der Bedeutungsinhalt einer Mitteilung oder Nachricht, die einen Sachverhalt ausdrückt und einem bestimmten Zweck dient. (Print: [1])
„Die Bedeutung einer Nachricht für den Empfänger (die einen Sachverhalt ausdrückt, einem Zweck dient oder eine Aktion auslöst) wird umgangssprachlich als Information bezeichnet. (Print: [2]))

Modell

Ein Modell ist ein zweckgebundes, vereinfachtes Abbild der Wirklichkeit. Modelle helfen, Informatiksysteme zu verstehen. (Print: [3])
Beispiele für Modelle sind z. B.:

- Objekte und Klassen,
- Blockschaltbild der Zentraleinheit oder
- Struktogramme

Begriffe für die Klassenstufe 7

Aktion

Mit Aktionen werden Objekte verwaltet, z. B. erzeugt, kopiert oder gelöscht.

Attribut

Attribute sind die Merkmale der Objekte. Attribute haben zu jeder Zeit konkrete Werte.

Bit

Ein Bit ist die kleinste Einheit der Datendarstellung und kann nur zwei mögliche Werte `(0,1)` annehmen. Diese können auf verschiedene Weise realisiert werden:

- optisch: (dunkel, hell)
- elektrisch: (aus, ein)
- magnetisch: (Nord, Süd)
- ...

Bus

Der Bus verbindet durch Kommunikationsleitungen die verschiedenen Bestandteile des Computers.

Byte

Ein Byte nennt man die Zusammenfassung von 8 Bit.
Es ist die Maßeinheit für die Kapazität von Speichermedien wie Festplatten. Größere Maßeinheiten sind KILOBYTE (KB), MEGABYTE (MB), GIGABYTE (GB) und TERRABYTE (TB). Wichtig: Kilo bedeutet hier $2^{10} = 1024$ und nicht 1000!

Chiffre

Chiffren sind Vorschriften zum Verschlüsseln von Nachrichten, damit diese nicht durch Unbefugte gelesen werden können. Eine einfache Chiffre ist die Cäsar-Verschlüsselung.

Code

Ein Code ist die „Übersetzung“ einer Nachricht in eine übertragbare Form. Ein bekannter Code ist der ASCII-Code.

Datei

Eine Datei enthält einen zusammengehörigen Datenbestand. Jede Datei hat typische Attribute, z. B. einen bestimmten Namen, eine bestimmte Größe, ein bestimmtes Dateiformat (Dateityp), u. a.

Dateityp

Der Dateityp beschreibt die Vereinbarung, nach der Daten in einer Datei gespeichert sind. Man unterscheidet ausführbare Dateien (z. B. Programme) und nichtausführbare Dateien (z. B. Dokumente).

EVA-Prinzip

Das Prinzip „Eingabe – Verarbeitung – Ausgabe“ ist ein in der Informatik typisches Prinzip. Es gilt nicht nur für die Programmierung, sondern auch in allen Anwendungen, z. B wird. in der Textverarbeitung zuerst ein neuer Text eingegeben, anschließend bearbeitet (geändert) und gestaltet (formatiert) und schließlich ausgegeben (auf einen externen Speicher oder Drucker).

Hardware

Als Hardware bezeichnet man alle mechanischen und elektronischen Bestandteile eines Computers bzw. eines Computersystems, also auch alle Peripheriegeräte und Netzwerkkomponenten. Hardware ist alles, was gegenständlich ist.

Objekt

Ein Objekt ist ein konkretes „Ding“. Jedes Objekt hat Attribute, die jeweils einen bestimmten Attributwert haben. Zur grafischen Darstellung von Objekten werden Rechtecke verwendet. In diesen werden notiert.

- der Name des Objektes (Kleinschreibung)
- die Namen der Attribute und deren aktueller Wert

Beispiele:

mein_fahrrad	
rahmenfarbe=	blau
radgroesse=	26 Zoll
...	

lackieren→

mein_fahrrad	
rahmenfarbe=	rot
radgroesse=	26 Zoll
...	

diagramm_sprecherwahl	
art=	Säule
titel=	Sprecherwahl
...	

diagrammart_aendern→

diagramm_sprecherwahl	
art=	Kreis
titel=	Sprecherwahl
...	

Operation

Mit Operationen werden Attributwerte von Objekten geändert.

Pfad

Ein Pfad bezeichnet den Speicherort einer Datei. Er enthält Laufwerksbezeichner, Verzeichnis, mögliche Unterverzeichnisse und den Dateinamen.

Prozessor

Der Prozessor oder Mikroprozessor ist das Herzstück des Computers. Er ist Bestandteil der Zentraleinheit. Der Prozessor verarbeitet die Daten.

Software

Software ist die Gesamtheit aller Programme und Daten, die bei der Arbeit mit dem Computer verwendet werden. Programme können in folgende Gruppen eingeteilt werden:

- SYSTEMSOFTWARE zur Steuerung der Hardware, zur Ausführung von Anwendungsprogrammen und zur Erstellung von Programmen (z. B. Betriebssystem, Treiber, Protokolle)
- ANWENDERSOFTWARE zur Lösung der Probleme des Anwenders (z. B. Textverarbeitungs-programme, Programmierumgebungen, Lernprogramme, Computerspiele).

Speicher

Ein Speicher ist ein Bestandteil des Computers, der Daten aufnehmen (einlesen), aufbewahren (sichern) und abgeben (auslesen) kann. Man kann unterscheiden zwischen:

- interne und externe Speicher
- Festwertspeicher und wiederbeschreibbare Speicher
- flüchtige und dauerhafte Speicher.

Verzeichnis

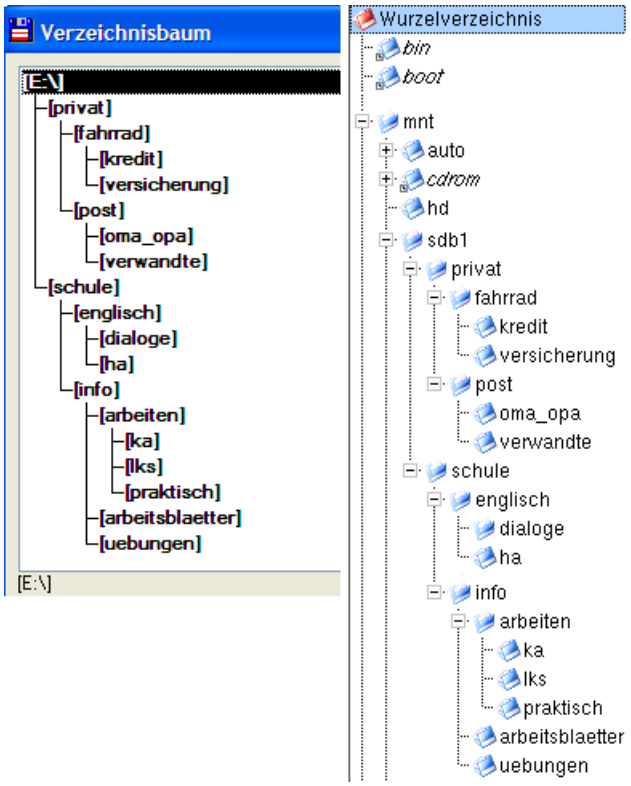
Dateien werden immer in Verzeichnissen (Ordnern) gespeichert. Mit Hilfe von Verzeichnissen kann das Speichern der Dateien auf einem Datenträger sinnvoll organisiert werden. Ein Verzeichnis hat immer einen Namen.

Verzeichnisbaum

Damit die große Anzahl von Dateien auf Speichern sinnvoll abgespeichert wird, kann man in jedem Verzeichnis Unterverzeichnisse anlegen. Damit entsteht ein sogenannter „Verzeichnisbaum“. Das Wurzelverzeichnis hat kein übergeordnetes Verzeichnis.

Verzeichnisstruktur

Eine Verzeichnisstruktur besteht aus dem Verzeichnisbaum und den darin enthaltenen Dateien. In der grafischen Veranschaulichung ist es sinnvoll, die Dateien in Klammern hinter das entsprechende Verzeichnis zu schreiben.



Zentraleinheit

Die Zentraleinheit, auch Zentrale Verarbeitungseinheit (ZVE, engl. CPU – Central Processing Unit) enthält als wichtige Bestandteile den Prozessor, den Hauptspeicher und Register. Diese Bestandteile sind untereinander durch den Bus und mit den Peripheriegeräten verbunden.

Begriffe für die Klassenstufe 8

Algorithmus

Ein Algorithmus ist eine Handlungs- bzw. Verarbeitungsvorschrift. Er besitzt folgende Eigenschaften:

- *Ausführbarkeit und Allgemeingültigkeit*:
  Er löst stets eine Menge von gleichartigen Problemen.
- *Wiederholbarkeit*:
  Er liefert für gleiche Eingabewerte immer das gleiche Endergebnis.
- *Eindeutigkeit*:
  Der nächste Schritt ist immer eindeutig festgelegt.
- *Endlichkeit (der Länge und der Ausführung)*:
  Er besteht aus einer endlichen Anzahl von Anweisungen und kommt immer zu einem Schluss.

Klasse

Objekte mit gleichen Attributen werden zu einer Klasse zusammengefasst. Dabei haben die Attribute einer Klasse jeweils einen Wertebereich. Die Objekte einer Klasse haben die gleichen Attribute, aber oft unterschiedliche Attributwerte aus dem jeweiligen Wertebereich. Merkmale einer Klasse sind die Attribute und die Methoden.
Beispiele: Darstellung der Klasse und eines Objekts der Klasse:

FAHRRAD	
rahmenfarbe:	blau, rot, gelb, ...
radgroesse:	20 Zoll, 24 Zoll, ...
...	
bremsen()	
lackieren()	
...	

ottos_fahrrad: FAHRRAD	
rahmenfarbe=	rot
radgroesse=	26 Zoll
...	
bremsen()	
lackieren(blau)	
...	

Methode

Methoden beschreiben das Verhalten der Objekte einer Klasse. In ihnen sind Operationen und Aktionen zusammengefasst.
Beispiele: Anwenden einer Methode auf ein Objekts der Klasse FAHRRAD

vorher

mein_fahrrad: FAHRRAD	
rahmenfarbe=	blau
radgroesse=	26 Zoll
...	
bremsen()	
...	

lackieren(rot)→

nachher

mein_fahrrad: FAHRRAD	
rahmenfarbe=	rot
radgroesse=	26 Zoll
...	
bremsen()	
...	

Problemlöseprozess

Jeder Problemlöseprozess besteht aus den folgenden vier wesentlichen Schritten:

- Beschreibung des Problems (Analyse)
- Auswahl eines geeigneten Beschreibungsmittels (Modellbildung)
- Umsetzung mit geeignetem Werkzeug (Implementierung)
- Test, Interpretation und Bewertung der Ergebnisse. (Test/Kritik)

Programm

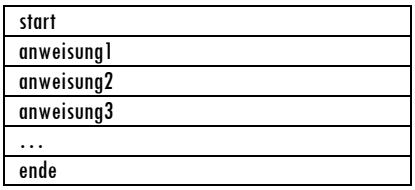
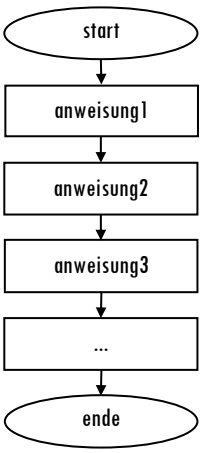
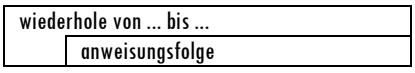
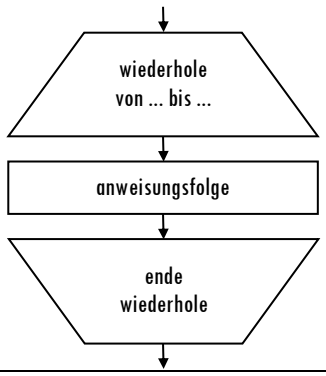
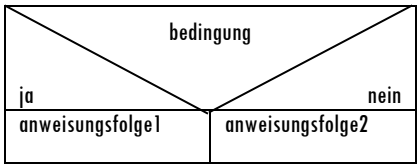
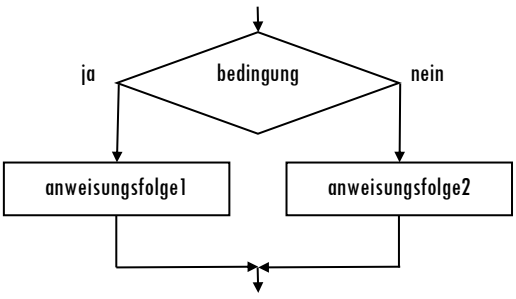
Als Programm bezeichnet man die Formulierung eines Algorithmus in einer für den Computer verständlichen Form. Programme können von Computern ausgeführt werden.

Programmstrukturen (Algorithmische Grundstrukturen)

In der Informatik gibt es folgende algorithmische Grundstrukturen:

- Folge
- Wiederholung
- Verzweigung

Diese Grundstrukturen können durch eine verbale Beschreibung, einen Programmablaufplan oder ein Struktogramm veranschaulicht werden.

<table>
<tr><th>Struktogramm</th><th>Programmablaufplan (PAP)</th></tr>
<tr><td colspan="2">Darstellung der Folge</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Darstellung der Wiederholung (Beispiel Zählschleife)</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Darstellung der Verzweigung</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

5.2 Aufgaben

Die Aufgaben sind exemplarisch und sollen verschiedene Anforderungsstufen in den einzelnen Lernbereichen verdeutlichen. Die Nutzbarkeit der Aufgaben hängt auch davon ab, welche Variante des Vorgehens für den Unterricht gewählt wurde. Bei der Auswahl wurde besonders auf unterschiedliche Arten der Schülertätigkeit geachtet, um zu analogen Aufgabenstellungen anzuregen. Deshalb sind reine Reproduktionsaufgaben bewusst nicht vertreten. Viele der Aufgaben sind auch ohne Computer lösbar und eignen sich damit als Hausaufgaben.
Die Aufgaben sind inhaltlich den einzelnen Lernbereichen der Jahrgangsstufen zugeordnet.

Aufgaben zu Klassenstufe 7 LB 1

Aufgabe A1:

Benenne die Bauteile und ordne sie durch Ankreuzen den Begriffen „Eingabe“, „Verarbeitung“ und „Ausgabe“ zu!

Bezeichnung Bauteil	Bild	E	V	A
				
				
				
				
				
				
				
				
				
				

Aufgabe A2:

Erstelle mit einem Zeichenprogramm ein Blockschaltbild eines Computers. Verwende unterschiedliche Farben für Geräte der Eingabe, Verarbeitung und Ausgabe. Trage folgende Begriffe in dein Bild ein: Tastatur, Maus, Zentraleinheit, Prozessor, Bus, interne Speicher, externe Speicher, RAM, ROM, Festplatte, USB-Stick, SD-Karte, DVD/CD-ROM, Bildschirm, Drucker. Stelle den Datenfluss in deinem Blockschaltbild durch Pfeile dar.

Als Vorbereitung auf diese Aufgabe sollte ein Entwurf des Blockschaltbildes zu Hause vorbereitet werden.

Aufgabe A3:

Schneide die einzelnen Schilder aus und entwickle mit ihnen eine Verzeichnisstruktur. Klebe die Zettel auf ein weißes Blatt Papier und ergänze mit Bleistift und Lineal den Verzeichnisbaum.
Verwende die leeren Schilder und ergänze eigene Beispiele.

Verzeichnisnamen	Dateinamen	
EUROPA	frankreich.doc	ungarn.xls
AMERIKA	polen.doc	china.doc
ASIEN	japan.htm	brasilien.htm
AUSTRALIEN	deutschland.xls	argentinien.doc
NORD	polen.xls	usa.htm
SUED	usa.xls	canada.htm
	guinea.xls	tunesien.doc
	kongo.doc	chile.xls
	russland.htm	
	mexiko.wdb	
	indien.xls	

Aufgabe A4:

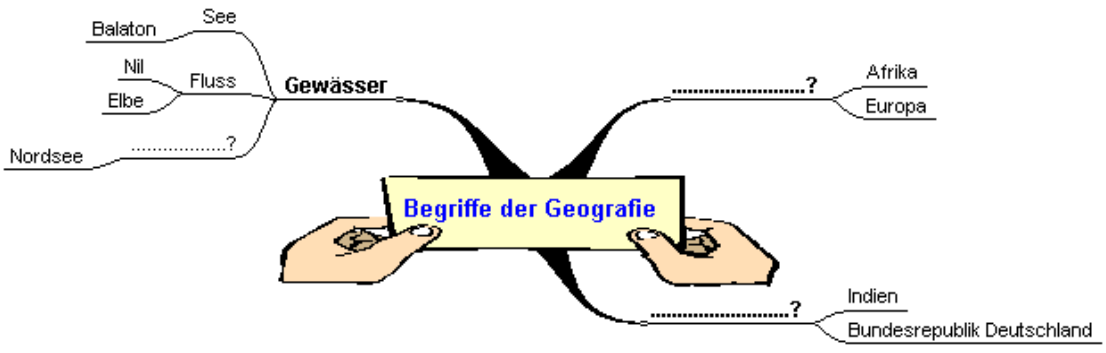
Auf einer Festplatte (Datenträger) sind die folgenden Dateien gespeichert:

apfel.doc	banane.xls	birne.txt	bohne.htm	erbse.fm3
kartoff.doc	ketschup.xls	kirsche.txt	kiwi.htm	kohl.doc
kohlrabi.txt	mais.wdb	mandarin.doc	moehre.htm	orange.txt
paprika.xls	pfirsich.txt	pflaume.xls	ruebe.fm3	senf.fm3

Plane eine mögliche Verzeichnisstruktur für dieses Beispiel. Verwende MINDESTENS EINMAL Unterverzeichnisse. Begründe deinen Entwurf.
Erstelle den Verzeichnisbaum auf deinem Datenträger und verschiebe die Dateien in die entsprechenden Unterverzeichnisse. Ordne anschließend die Dateien aus dem Netzlaufwerk in deine Verzeichnisstruktur ein.

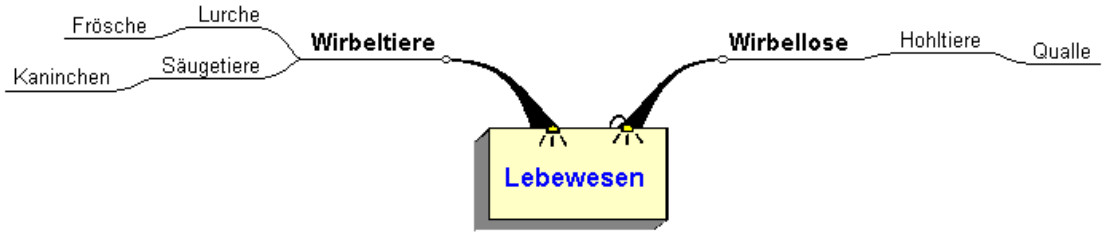
Aufgabe A5:

Ergänze die gegebene Struktur mit folgenden Begriffen: Land, Kontinent, Meer, Frankreich, Australien, Ostsee



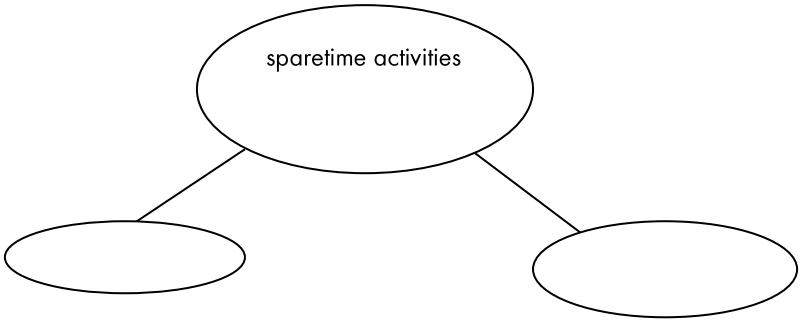
Aufgabe A6:

Erstelle aus folgenden Begriffen der Biologie eine Mind Map, strukturiere nach Gattung, Art, Unterart und Vertreter. Setze die gefundene Struktur in einem Verzeichnisbaum in deinem Schülerordner um.
Wirbeltiere, Frösche, Säugetiere, Lurche, Wirbellose, Lebewesen, Hohltiere, Kaninchen, Qualle
Lösungsvorschlag:



Aufgabe A7:

Ordne folgende Vokabeln den zeitlichen Oberbegriffen zu und stelle sie in einer Mind Map (NETWORK) dar. Finde weitere passende Vokabeln.
sparetime activities, to run, surfing, skating, on holidays, beach, volleyball, sea, lake, sunshine, icecreme, in the afternoon, meet friends, go shopping, go to the cinema, go climbing, do sports, …



Aufgabe A8:

Ordne folgende Begriffe sinnvoll in die Tabelle ein: Rose, Gurke, Kirsche, Tulpe, Nelke, Tanne, Linde, Anemone, Apfel, Gerbera, Schneeglöckchen, Erdbeere, Pflaume, Birke, Buche, Weide, Kartoffel, Tomate. Ergänze frei bleibende Zellen durch selbst gewählte Begriffe.

Aufgaben zu Klassenstufe 7 LB 2

Aufgabe A9:

Bestimme mindestens vier Eigenschaften deines Schülertisches (Abmessungen, Farbe, ...). Stelle den Tisch als Objekt mit seinen Attributen grafisch dar. Beschreibe weitere Möbelstücke des Klassenzimmers.
Lösungsvorschlag:

mein_schuelertisch	
hoehe=	72 cm
arbeitsflaeche=	9600 cm²
farbe=	beige
beinzahl=	4

Aufgabe A10:

Schneide aus einem Prospekt für Digitalkameras zwei Angebote heraus. Erstelle für beide Kameras eine grafische Darstellung mit mindestens vier vergleichbaren Attributen.
Lösungsvorschlag:

camedia_350_zoom	
hersteller=	Olympus
aufloesung=	3,2 Megapixel
preis=	187 €
masse=	215 g

powershot_a_70	
hersteller=	Canon
aufloesung=	3,2 Megapixel
preis=	225€
masse=	305 g

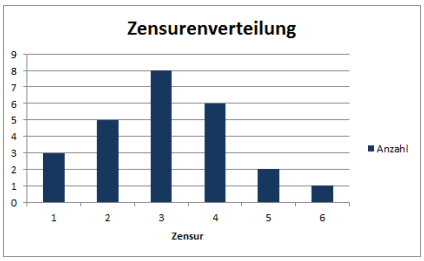
Aufgabe A11:

Gliedere den folgenden Text in Überschrift, Zutaten und Anweisung. Gestalte damit eine Kochbuchseite.
FalscheEierscheckeTeig200gMehl100gZucker75gMarina1Ei1Eigelb1/2PäckchenBackpulverAllesverrührenundeinegefetteteFormgebenBelag500gQuark150gZucker1kleineTasseÖl1PäckchenPudding1Ei3Eigelballesmiteinemhalben LiterMilchverrührenDieMasseaufdenTeiggebenundetwa45Minutenbacken(bisderRandleichtbraunwird)FürdieSchecke4Eiweißsteifschlagenund4EsslöffelZuckervorsichtigdarunterhebenAllesgleichmäßigaufdenKuchenverteilenundnochmals15bis20MinutenbackenGutenAppetit.

Aufgaben zu Klassenstufe 8 LB 1

Aufgabe A12:

Wiederholung: Leite aus dem Diagramm der Zensurenverteilung die zugehörige Objektdarstellung ab.



diagramm_zensurenverteilung	
art=	Säule
daten=	A1:B7
x_achse=	Zensur
y_achse=	Anzahl
titel=	Zensurenverteilung
legende=	sichtbar
name=	Diagramm_Zensurenverteilung

Aufgabe A13:

Ergänze die folgenden Beispiele und finde jeweils eine Methode (Klasse) und eine Operation (Objekt)

Beispiel aus der Schule

Klasse

.....	
name:	Archimedes, ... , Zuse
vorname:	Anton, ..., Xaver
geschlecht:	m,w
masse:	20 bis 150 kg
…	

Objekt

paul_mueller: SCHUELER	
name=	.....
vorname=	.....
geschlecht=	.....
masse=	.....
…	

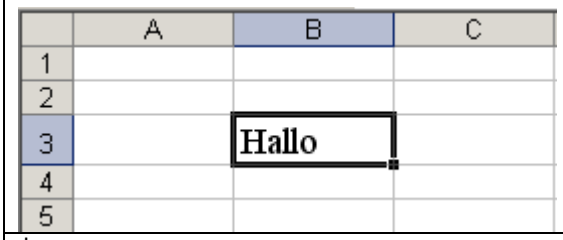
Beispiel aus der Tabellenkalkulation

Klasse

.....	
inhalt:	13, =B4+B5, Hallo
zahlendarstellung:	Währung in €, ...
textdarstellung:	Verdana 12pt Standard, ...
hintergrund:	gelb, ...
rahmen:	Linie doppelt rot unten, …
…	

Objekt

b3: ZELLE



datentyp=	.....
schriftgroesse=	.....
ausrichtung=	.....
schriftart=	.....
…	

Beispiel aus der Textverarbeitung

Klasse

.....	
schriftschnitt:	fett, kursiv…
schriftgroesse:	2..128 Punkt
schriftfarbe:	rot,.blau, ...
schriftart:	Arial, Times New Roman, ...
…	

Objekt

e: ZEICHEN	
schriftschnitt=	.....
schriftgroesse=	.....
schriftfarbe=	.....
schriftart=	.....
…	

Beispiel aus der Vektorgrafik

Klasse

.....	
position:	x1, y1; x2, y2
linienstaerke:	2..128 Punkt
linienfarbe:	rot,.blau, ...
linienart:	Volllinie, Strichlinie, Strich Punkt-Linie, ...
…	

Objekt

mittellinie: LINIE	
position=	20, 30, 40, 30
linienstaerke=	.....
linienfarbe=	.....
linienart=	.....
…	

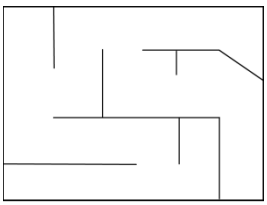
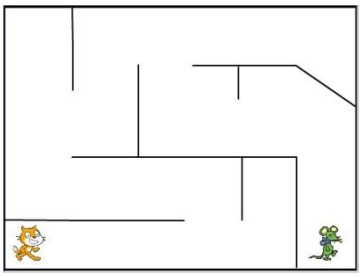
Aufgaben zu Klassenstufe 8 LB 2

Aufgabe A14 (zur Arbeit mit Scratch):

Katze soll immer geradeaus laufen. Ist eine Hürde im Weg, soll sie darüberspringen und danach seinen alten Weg fortsetzen. Überlege, welchen Sensor Katze benötigt. Erstelle ein Programm und teste es (auch an verschiedenen Bühnen, beispielsweise an denen deiner Mitschüler). Beschreibe jeweils, was beim Ablauf deines Programms passiert. Finde Ursachen für aufgetretene Fehler.

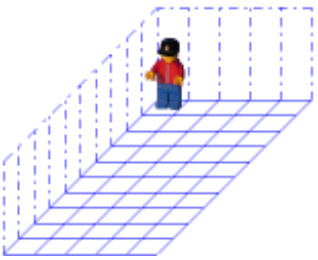
Aufgabe A15 (zur Arbeit mit Scratch):

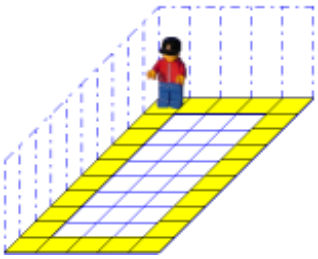
Gegeben ist folgende Bühne mit einem Labyrinth und die Ausgangssituation:
Katze soll vom Startpunkt des Labyrinths die Maus finden. Erstelle die Bühne und das zugehörige Programm. Gestalte es interaktiv z.B. durch die Steuerung mittels Pfeiltasten.

Bühne:	Ausgangssituation:
	

Aufgabe A16 (zur Arbeit mit Karol the Robot):

Karol soll eine beliebige leere Welt umrunden und sich den Weg wie Hänsel und Gretel „merken“. Statt Brotkrumen setzt Karol Marken auf jedes betretene Feld. Karol startet bei seiner Erkundung der Welt aber immer hinten links mit dem Blick nach vorne. Kommt Karol auf ein markiertes Feld, soll er stehen bleiben.





Lösungsvorschlag:

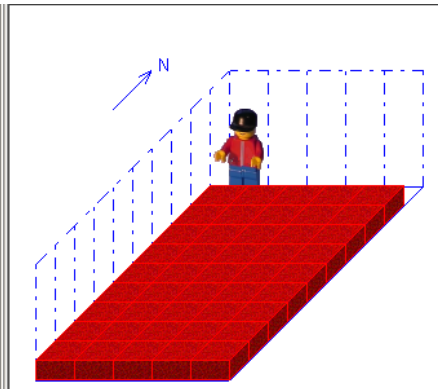
```
solange NichtIstMarke tue
  solange NichtIstWand tue
    MarkeSetzen
    Schritt
  *solange
  LinksDrehen
*solange
```

solange NichtIstMarke	
	solange NichtIstWand
	MarkeSetzen
	Schritt
	LinksDrehen

Aufgabe A17 (zur Arbeit mit Karol the Robot):

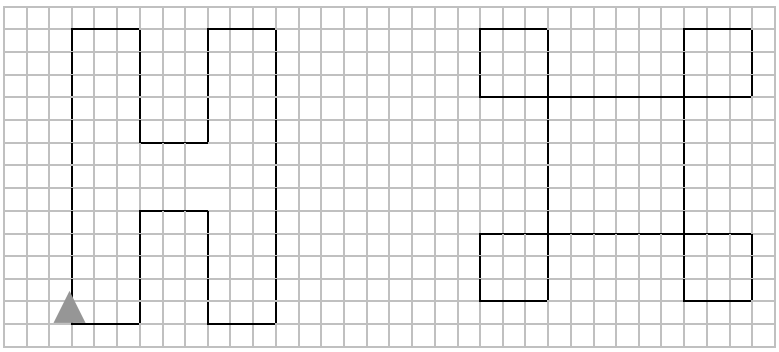
Erstelle zu folgendem Programm von Karol das Struktogramm (ohne Computer!). Beschreibe, was Karol beim Ausführen des Programms tut. An welcher Stelle in der dargestellten Welt bleibt Karol bei einem Rucksackfassungsvermögen von 35 Ziegeln stehen?

```
wiederhole immer
  solange NichtIstWand tue
    wenn NichtIstZiegel dann
      Schritt
      RechtsDrehen
      Schritt
      RechtsDrehen
    sonst
      wenn NichtIstVoll dann
        Aufheben
        Schritt
      sonst
        MarkeSetzen
        Beenden
      *wenn
    *wenn
  *solange
  LinksDrehen
  wenn NichtIstVoll dann
    Aufheben
    Schritt
  sonst
    MarkeSetzen
    Beenden
  *wenn
  LinksDrehen
*wiederhole
```



Aufgabe A18 (zur Arbeit mit Logo):

Der Igel soll die gegebenen Figuren zeichnen. Notiere den Ablauf für die Steuerung des Igels unter den Bildern. Lege beim zweiten Beispiel den Startpunkt des Igels selbst fest.



Aufgabe A19 (zur Arbeit mit Logo):

Simuliere die Arbeit des Igels und arbeite die Folge ab. Zeichne die dabei entstehende Figur auf kariertes Papier. Gib auch die Startposition (grün) und die Endposition (rot) des Igels an.

```
lerne figur1
      vw 100 re 90 vw 100 rw 100 li 90 rw 100
      sh re 90 vw 10 li 90 sa
      vw 90 re 90 vw 90 rw 90 li 90 rw 90
      sh re 90 vw 10 li 90 sa
      vw 80 re 90 vw 80 rw 80 li 90 rw 80
      sh re 90 vw 10 li 90 sa
      vw 70 re 90 vw 70 rw 70 li 90 rw 70
      sh re 90 vw 10 li 90 sa
      vw 60 re 90 vw 60 rw 60 li 90 rw 60
      sh re 90 vw 10 li 90 sa
ende
```

Aufgabe A20 (zur Arbeit mit Logo):

Simuliere die Arbeit des Igels und arbeite die Wiederholungen ab. Zeichne die dabei entstehenden Figuren, auf kariertes Papier. Gib auch jeweils die Start-Position (grün) und die Ende-Position (rot) des Igels an.
Begründe die unterschiedlichen Ergebnisse.

```
lerne figur2                    lerne figur3
      wh 3 [vw 100 li 120]            wh 3 [li 120 vw 100]
ende                            ende
```

Aufgabe A21 (zur Arbeit mit Logo):

Analysiere die Verzweigung.
Notiere den Aufruf der Prozedur, die notwendig ist, um ein Dreieck zu zeichnen.
Nenne die Figur, die beim Aufruf von **figuren 0** gezeichnet wird.
Ändere die Prozedur so, dass nur beim Aufruf von **figuren 6** ein Sechseck gezeichnet wird. Im anderen Fall soll ein Dreieck gezeichnet werden.

```
lerne figur1
      wh 3 [vw 100 li 120]
ende

lerne figur2
      wh 4 [vw 100 li 90]
ende

lerne figuren :e
      wennsonst :e<4 [figur1] [figur2]
ende
```

6 Fachbegriffe und fachliche Grundlagen für den Lehrer

6.1 Ausgewählte Fachbegriffe

Der Zugang zur Informatik kann am Anfang durch das Auftreten einer Vielzahl von Begriffen beeinträchtigt werden. Es erscheint gerade aus diesem Grund dringend geboten, als Fachlehrer Begriffsbildung gezielt zu betreiben. Das durch die Varianten angeregte didaktisch-methodische Vorgehen und die in der Folge getroffene Auswahl von fachlichen Themen und Gegenständen darf nicht dazu verleiten, eine unnötige Anzahl von Begriffen im Unterricht zu verwenden. Unbedingt zu vermittelnde Begriffe sind im Lehrplan festgeschrieben und im Kapitel 5 mit einer schülergemäßen Definition aufgeführt. In Abhängigkeit vom eigenen Vorgehen sind den Schülern gegebenenfalls weitere Begriffe zu vermitteln. Das will der folgende Abschnitt durch die **Zusammenstellung weiterer Begriffe (für den Lehrer – nicht als Merksatz für Schüler) unterstützen.**

Dieses Kapitel kann das Studium von Fachliteratur durch den Lehrer nicht ersetzen. Es soll Anregung und Hilfe sein, sich mit bestimmten Themen intensiver zu beschäftigen und den Zugang zu Fachbegriffen erleichtern.

Für den Schüler muss immer erkennbar sein, welche Begriffe sicher verfügbar sein müssen und welche den Status der Information besitzen. Die exakte Arbeit mit Begriffen kennzeichnet den fachlich soliden Unterricht und hilft dem grundlegenden Verständnis der Informatik und ihren Anwendungen. In diesem Zusammenhang sollte auch dem sich mitunter schnell prägenden „Computerslang“ entgegengewirkt werden. Die Schüler sollen erfahren, dass die Informatik ein etabliertes Fachgebiet ist, das bei aller Aktualität von neuen Entwicklungen in seinem Kern ein gesichertes begriffliches Gerüst besitzt.

Aus dieser Sicht sind die folgenden Begriffe angegeben bzw. in ihrer Darstellung bezüglich Kapitel 5 erweitert.

Das bedeutet auch, dass die **Programmiersprachen nicht in dem hier dargestellten Umfang zu behandeln** sind. Das ist weiterführendes Material.

Algorithmische Grundstrukturen

Es gibt folgende algorithmische Grundstrukturen:

- Folge (Sequenz)
- Wiederholungen (Zyklen)
  - Zählschleife
  - Schleife mit vorangestelltem Test
  - Schleife mit nachgestelltem Test
- Verzweigungen
  - Einseitige Auswahl
  - Zweiseitige Auswahl (Alternative)
  - Mehrfachauswahl

Für den Schüler vereinfachte Struktogramme und Programmablaufpläne sind im Kapitel 5 angegeben. Nachfolgend werden für den Lehrer Struktogramme (auch: Nassi-Shneidermann-Diagramm) (Print: [5]) und Quelltextfragmente für diese Grundstrukturen in verschiedenen Programmiersprachen angegeben:

FOLGE:

Struktogramm:

Allgemein:

Anweisung 1
Anweisung 2
Anweisung 3
...
...
Anweisung n

Beispiel:

vorwärts 100
rechts 120
vorwärts 100
rechts 120
vorwärts 100
rechts 120

Logo: Eine Folge von Grundprozeduren wird nur durch je ein Leerzeichen getrennt:

```
def "dreieck [[a] [vw :a re 120 vw :a re 120 vw :a re 120]]
```

Objekt Pascal (Delphi): Anweisungen werden durch Semikolon voneinander getrennt.

JavaScript: Anweisungen werden mit einem Semikolon abgeschlossen.

php: Anweisungen werden mit einem Semikolon abgeschlossen.

WICHTIGE VORBEMERKUNG: In allen folgenden Struktogrammen kann anstelle einer einzelnen Anweisung jeweils ein Block aus mehreren Anweisungen bzw. jede andere algorithmische Grundstruktur stehen.

ZÄHLSCHLEIFE:

Struktogramm:

Allgemein:

für ... bis ... tue
Anweisung

Beispiel:

für i := 1 bis 10 tue	
	su := su + i

Logo:

```
wiederhole <anzahl> [ <anweisung bzw. -folge>]
wh 3 [vw 100 re 120]
```

Pascal:

```
for <anfang> to <ende> do <anweisung bzw. -folge>;
for i := 1 to 10 do su := su + i;
```

JavaScript:

```
for (<anfang>; <ende>; <schrittweite>) {<anweisung bzw. -folge>}
for (var i=0; i<=10; i++) {su = su + i;}
```

php:

```
for (<anfang>; <ende>; <schrittweite>) {<anweisung bzw. -folge>}
for (var $i=0; $i<=10; $i++) {$su = $su + $i;}
```

SCHLEIFE MIT VORANGESTELLTEM TEST:

Struktogramm:

Allgemein:

solange (Bedingung)
Anweisung

Beispiel:

solange i <= 10
su := su + i
i := i+1

Pascal:

```
while <bedingung> do <anweisung bzw. -folge>;
while i <= 10 do begin su := su + i; i := i + 1 end;
```

JavaScript:

```
while (<bedingung>) do {<anweisung bzw. -folge>}
while (i<=10) do {su = su + i; i++;}
```

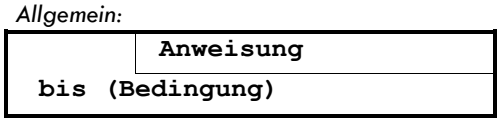
php:

```
while (<bedingung>) do {<anweisung bzw. -folge>}
while ($i<=10) do {$su = $su + $i; $i++;}
```

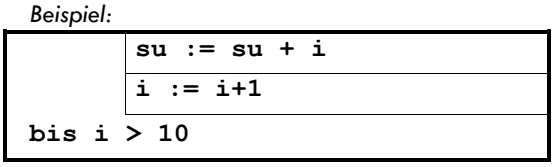
SCHLEIFE MIT NACHGESTELLTEM TEST:

Struktogramm:

Allgemein:



Beispiel:



Pascal:

```
repeat <anweisung bzw. -folge> until <bedingung>;
repeat su := su + i; i := i + 1 until i > 10;
```

JavaScript:

```
do {<anweisung bzw. -folge>} while (<bedingung>)
do {su = su + i; i++;} while (i<=10)
```

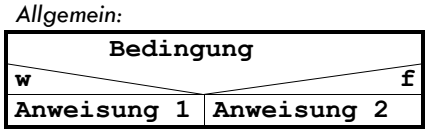
php:

```
do {<anweisung bzw. -folge>} while (<bedingung>)
do {$su = $su + $i; $i++;} while ($i<=10)
```

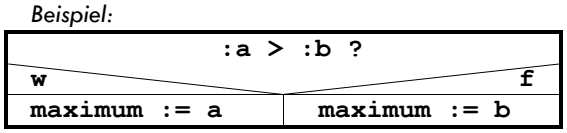
VERZWEIGUNG

Struktogramm:

Allgemein:



Beispiel:



Logo:

```
wennsonst <bedingung> [<wert1>][ <wert2>]
wennsonst :a > :b [rg :a][rg :b]
```

Pascal:

```
if <bedingung> then <anw. bzw. -folge> else <anw. bzw. -folge>;
if a > b then maximum := a else maximum := b;
```

JavaScript:

```
if (<bedingung>) {<anw. bzw. -folge>} else {<anw. bzw. -folge>}
if (a>b) {maximum = a;} else {maximum = b;}
```

php:

```
if (<bedingung>) {<anw. bzw. -folge>} else {<anw. bzw. -folge>}
if ($a>$b) {$maximum = $a;} else {$maximum = $b;}
```

MEHRFACHAUSWAHL

Struktogramm:

Allgemein:

Fallauswahl			
Fall 1	Fall 2	...	Fall n
Anw. 1	Anw. 2	...	Anw. n

Beispiel:

tag =			
0	1	...	6
wtag = "Sonntag"	wtag = "Montag"	...	wtag = "Sonnabend"

Pascal:

```
case <selektor> of
     <wert 1> : <anw. bzw. -folge>
     <wert 2> : <anw. bzw. -folge>
     ...      : ...;
     <wert n> : <anw. bzw. -folge>
end;
case tag of
    case 0 : wtag = "Sonntag";
    case 1 : wtag = "Montag";
    ...    : ...;
    case 6 : wtag = "Sonnabend";
end;
```

JavaScript:

```
switch (<selektor>)
 {
    case <wert 1> : <anw. bzw. -folge>   break;
    case <wert 2> : <anw. bzw. -folge>   break;
    ...           : ...                  break;
    case <wert n> : <anw. bzw. -folge>
 }
switch (tag)
 {
    case 0 : wtag = "Sonntag";    break;
    case 1 : wtag = "Montag";     break;
    ...    : ...;                 break;
    case 6 : wtag = "Sonnabend";
 }
```

php:

```
switch (<selektor>)
 {
    case <wert 1> : <anw. bzw. -folge>   break;
    case <wert 2> : <anw. bzw. -folge>   break;
    ...           : ...                  break;
    case <wert n> : <anw. bzw. -folge>
 }
switch ($tag)
 {
    case 0 : $wtag = "Sonntag";    break;
    case 1 : $wtag = "Montag";     break;
    ...    : ...;                  break;
    case 6 : $wtag = "Sonnabend";
 }
```

Algorithmus

Ein Algorithmus ist eine Vorschrift zur Verarbeitung von Daten, die so formuliert ist, dass er von einer Maschine (auch von einem Menschen) ausgeführt werden kann. Jeder Algorithmus hat folgende Eigenschaften:

allgemeingültig	löst eine Menge von gleichartigen Problemen (Ausführbarkeit, Allgemeingültigkeit)
determiniert	bei gleichen Eingaben und Bedingungen gleiche Resultate (Wiederholbarkeit)
deterministisch	zu jedem Zeitpunkt der Ausführung nur eine Möglichkeit der Fortsetzung (Eindeutigkeit)
finit	Beschreibung mit endlicher Länge (Endlichkeit)
terminiert	für jede Eingabe entsteht nach endlich vielen Schritten ein Resultat (Endlichkeit)

Diese Definition bezeichnet man auch als „naiven“ Algorithmusbegriff. Mathematisch gesehen, ist ein Algorithmus eine Abbildung von einer Menge zulässiger Eingabedaten in die Menge der Ausgabedaten. Eine exakte mathematische Definition des Algorithmusbegriffs kann z. B. mit Hilfe einer Turingmaschine angegeben werden. Die Angabe von verschiedenen gleichwertigen Definitionen des Begriffes Algorithmus ist ein wichtiges Gebiet der Theoretischen Informatik (Print: [6]).

Algorithmen können auf verschiedene Art und Weise beschrieben werden:

- mit Worten der Umgangssprache (als Folge von Befehlen)
- mittels Struktogrammen
- mittels Programmablaufplänen
- mit Worten der Umgangssprache als Beschreibung funktionaler Zusammenhänge
- mittels miteinander verketteter Funktionen
- mit Automaten
- mit Programmen in einer Maschinensprache
- mit Programmen in einer höheren Programmiersprache

Anwendung

Eine Anwendung ist eine neue Arbeitsweise zum Lösen von gleichartigen Problemen und Aufgaben mit Hilfe des Computers. Jede Anwendung gliedert sich in ein Managementsystem und die dazugehörigen Daten. Im Managementsystem sind die Methoden der Anwendung implementiert.

Es entstanden sogenannte „Standardanwendungen“ wie:

- Textverarbeitung (Bearbeitung von Texten mit dem Computer)
- Tabellenkalkulation (Durchführung komplexer Berechnungen in Tabellen mit dem Computer)
- Datenbanken (Verwaltung und schnelle Auswertung großer Datenmengen mit dem Computer)
- Bildbearbeitung (Bearbeiten vorhandener Pixel- und Vektografiken mit dem Computer)

Für jede dieser Standardanwendungen existieren verschiedene Managementsysteme, denen jedoch jeweils gleiche Prinzipien zu Grunde liegen.

Baum

Ein Baum ist eine rekursive Datenstruktur. Er besteht entweder aus einem einzelnen Knoten oder einem Knoten und einer Liste von Unterbäumen. Es gibt genau einen Knoten, der keinen Vorgänger hat. Diesen bezeichnet man als Wurzel des Baumes. Knoten, die keinen Nachfolger haben, bezeichnet man als Blätter des Baumes. Bäume sind somit eine Struktur, die sehr gut zur Darstellung von Hierarchien geeignet ist. In Kapitel 5 sind zwei Verzeichnisbäume als Beispiele angegeben.

Wenn jeder Knoten eines Baumes genau zwei Nachfolgeknoten hat, so spricht man von einem binären Baum. Diese spielen in der Informatik, insbesondere bei Such- und Sortierverfahren, eine wichtige Rolle.

Betriebssystem

Unter einem Betriebssystem versteht man die Gesamtheit aller Programme, die die Ausführung der Anwendungsprogramme, die Verteilung der Betriebsmittel und die Aufrechterhaltung des Betriebs sichern. Wichtige Betriebssysteme sind u. a. Windows, Unix, Linux, Mac OS, Android, iOS.

Binärdarstellung

Jede Darstellung von Daten, die sich eines Alphabets bedient, das nur aus zwei Zeichen besteht, nennt man Binärdarstellung. Ein wichtiges Beispiel für eine solche Darstellung aus zwei Zeichen ist das Morsealphabet. Es besteht aus den beiden Zeichen „Punkt“ und „Strich“.

Datei (engl. file)

In einer Datei werden die unter einem bestimmten Gesichtspunkt zusammengehörenden Daten zum Abspeichern auf einem externen Datenträger zusammengefasst. Auch Programme werden in Dateien (genauer: in „ausführbaren“ Dateien) gespeichert. Wichtige Attribute von Dateien sind u. a.: Dateityp, Dateigröße, Änderungsdatum, Rechte (Lese-, Schreib- und Ausführungsrecht für den Eigentümer, für eine bestimmte Gruppe bzw. für alle Nutzer).

Dateityp

Jede konkrete Datei hat als Objekt der Klasse „Datei“ einen Namen und eine ganze Reihe von Attributen wie „Größe“, „ausführbar“, u. a. Der Dateityp (oder das Dateiformat) bestimmt die Form der Abspeicherung der Daten in der Datei. Je nach Inhalt der Datei unterscheidet man z. B. Dateien mit formatiertem Text, Grafik-Dateien, Sounddateien. Das Managementsystem einer Reihe von Anwendungen vergibt beim Abspeichern der Daten automatisch eine Dateierweiterung (z. B. vergibt Staroffice unter Linux die Erweiterungen „sxw“ bzw. „sxc“ für Text- bzw. Kalkulationsdateien). Viele Anwendungsprogramme enthalten heute Filter, die auch das Einlesen „fremder“ Dateiformate ermöglicht. Außerdem sind einige plattformunabhängige Austauschformate wie „rtf“, „pdf“ u. a. entstanden. Nachfolgend werden einige Beispiele angegeben:

Dateityp	**Beschreibung**
*.exe, *.com	ausführbare Dateien (Programme)
*.txt	einfache Text-Dateien
*.rtf, *.docx, *.sxw, *.ods	Text-Dokumente, die auch Tabellen und Bilder enthalten können
*.ps, *.pdf	plattformunabhängiges Format für Dokumente, die sowohl Text, Tabellen, Grafiken und Bilder enthalten können.
*.xlsx, *.sxc	Kalkulationstabellen
*.html, *.htm	Hypertextdokumente
*.tif; *.gif, *.jpg, *.png	Pixelgrafik
*.ai, *.eps, *.svg	Vektorgrafik
*.avi, *.mpeg, *.mov, *.wmv, *.mp4	Videodateien
*.wav, *.mp3, *.aac, *.wma	Soundateien

Dezimalsystem

Das Dezimalsystem ist ein Positionssystem zur Zahldarstellung mit der Basis 10. Folglich stehen zur Zahldarstellung die Ziffern 0 bis 9 zur Verfügung. Nachfolgend ist ein Beispiel angegeben:
$(1204.05)_{10} = 1 * 10^3 + 2 * 10^2 + 0 * 10^1 + 4 * 10^0 + 0 * 10^{-1} + 5 * 10^{-2}$

Dualsystem

Das Dualsystem ist ein Positionssystem zur Zahldarstellung mit der Basis 2. Damit stehen zur Zahldarstellung nur die Ziffern 0 und 1 zur Verfügung. Nachfolgend werden einige Beispiele angegeben und in das Dezimalsystem umgerechnet:

$(111.1)_2 = 1 * 2^2 + 1 * 2^1 + 1 * 2^0 + 1 * 2^{-1} = 4 + 2 + 1 + 0.5 = (7.5)_{10}$

$(1101.01)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2} = 8 + 4 + 1 + 0.25 = (13.25)_{10}$

Hexadezimalsystem

Das Hexadezimalsystem ist ein Positionssystem zur Zahldarstellung mit der Basis 16. Zur Zahldarstellung stehen die Ziffern 0 bis 9 und die Buchstaben A (für 10), B (für 11) ... F (für 15) zur Verfügung.
Nachfolgend werden einige Beispiele angegeben und in das Dezimalsystem umgerechnet:

$(1A.1)_{16} = 1 * 16^1 + 10 * 16^0 + 1 * 16^{-1} = 16 + 10 + 0.0625 = (26.0625)_{10}$

$(FF)_{16} = 15 * 16^1 + 15 * 16^0 = 240 + 15 = (255)_{10}$

Informatiksystem

Ein Informatiksystem ist die spezifische Zusammenstellung von Hardware, Software und Netzverbindungen zur Lösung von Anwendungsproblemen.

Klasse

Eine Klasse ist ein „Bauplan“ für gleichartig strukturierte Objekte. Sie hat einen Namen und wird beschrieben durch ihre Methoden und ihre Attribute, mit jeweils zugehörigem Attributwertebereich. (Weitere Erläuterungen siehe Kapitel 6.2)

Konvertierung

Die Umrechnung einer Zahldarstellung aus einem Positionssystem zur Basis b_1 in ein Positionssystem zur Basis b_2 bezeichnet man als Konvertierung. Zur Umrechnung aus einem Positionssystem zur Basis $b_1 \neq 10$ in das Dezimalsystem kann man das sogenannte HORNER-Schema (siehe Mathematik) nutzen. Wichtig ist dabei: Der ganze und der gebrochene Anteil einer Zahldarstellung werden getrennt umgerechnet.

BEISPIEL: $(1101.01)_2 = (...)_{10}$

Ganzer Anteil: 1101

	1		1		0		1
	0	↓1+0	2	↓1+2	6	↓0+6	12 ↓1+12
2	1	↗1*2	3	↗3*2	6	↗6*2	**13**

Gebrochener Anteil: 0.01

	1		0		0	
	0	↓1+0	0.5	↓0+0.5	0.25	↓0+0.25
0.5	1	↗1*0.5	0.5	↗0.5*0.5	**0.25**	

Gesamtergebnis: $(1101.01)_2 = (13.25)_{10}$

Zur Umrechnung aus dem Dezimalsystem in ein Positionssystem zur Basis b_2 muss der ganze Anteil wiederholt durch die neue Basis dividiert und der gebrochene Anteil wiederholt mit der neuen Basis multipliziert werden:

BEISPIEL: $(13.25)_{10} = (...)_2$

13	:	2	=	6Rest **1**		0.25	*	2	=	**0**.5 ↓
6	:	2	=	3Rest **0**		0.50	*	2	=	**1**.0
3	:	2	=	1Rest **1**						
1	:	2	=	0Rest **1** ↑						

Gesamtergebnis: $(13.25)_{10} = (1101.01)_2$

Logische Verknüpfungen

In der Aussagenlogik hat jede Aussage einen Wahrheitswert. Dieser kann „wahr“ oder „falsch“ sein. Wichtige Aussagenverknüpfungen zweier Aussagen sind die „und“-Verknüpfung (dargestellt durch + oder $\wedge$ oder &) und die „oder“-Verknüpfung (dargestellt durch . oder $\vee$ oder |). Außerdem gibt es zu jeder Aussage „a1“ eine Aussage „nicht a1“ (dargestellt durch - oder $\neg$). Die folgenden Tabellen liefen die Wahrheitswerte der Verknüpfungen:

A	$\neg$ A
wahr	falsch
falsch	wahr

A	B	A $\wedge$ B
wahr	wahr	wahr
wahr	falsch	falsch
falsch	wahr	falsch
falsch	falsch	falsch

A	B	A $\vee$ B
wahr	wahr	wahr
wahr	falsch	wahr
falsch	wahr	wahr
falsch	falsch	falsch

Methode

Methoden beschreiben das Verhalten einer Klasse. Dabei kann man unterscheiden zwischen Operationen und Aktionen. Mit Operationen werden Attributwerte von Objekten geändert, mit Aktionen werden Objekte verwaltet (erzeugt, kopiert, gelöscht).

Objekt

Ein Objekt ist eine identifizierbare Einheit der Umwelt oder des Denkens. Alle Objekte, die aus einer Klasse erzeugt werden, besitzen die gleichen Attribute (aber i. Allg. unterschiedliche Attributwerte) und verfügen über die gleichen Methoden. (weitere Erläuterungen siehe Kapitel 6.2)

Problemlöseprozess

Jeder Problemlöseprozess besteht aus den folgenden vier Schritten:

- Problemanalyse (Beschreibung des Problems)
- Modellbildung (Auswahl eines geeigneten Modellbeschreibungsmittels)
- Algorithmierung/Implementierung (Umsetzung mit geeignetem Werkzeug)
- Test/Kritik (Interpretation der Ergebnisse und Bewertung in der Realität)

Prozessbegleitend sind die einzelnen Schritte und Teilergebnisse festzuhalten (Dokumentation).

Problemlösungsstrategie

In der Informatik gibt es verschiedener Problemlösungsstrategien:

Backtracking: Ein Verfahren, bei dem man versucht, eine Teillösung systematisch zu einer Gesamtlösung auszubauen. Geht es an einer Stelle nicht weiter (Sackgasse), so geht man einen oder mehrere Teilschritte zurück und versucht erneut, eine Gesamtlösung zu finden. Ein Beispiel dafür ist das sogenannte „Damenproblem“ (Print: [7]).

Rekursion: (lat. recurrere – zurücklaufen) Das zu lösende Problem wird auf das gleiche Problem mit veränderten Parametern zurückgeführt (Rekursionsschritt) bis das Problem gelöst werden kann (Abbruchbedingung).

BEISPIEL:

Zählen der Elemente in einer Liste:

```
el_anzahl(liste) = 0 ,    wenn die Liste leer ist (Abbruchbedingung)
el_anzahl(liste) = 1 + el_anzahl(liste_ohne_erstes_element)
                                                    (Rekursionsschritt)
```

Teile und herrsche: (engl. Divide-and-conquer) Das zu lösenden Problem wird in zwei oder mehr möglichst gleichgroße Teilprobleme derselben Art wie das Ausgangsproblem zerlegt, die unabhängig voneinander gelöst werden. Diese Teilproblemlösungen werden zu einer Lösung des Gesamtproblems zusammengefügt. Ein Beispiel dafür ist das Sortierverfahren „Quicksort“. (Print: [7])

Programm

Ein Programm ist die Formulierung eines Algorithmus in einer für den Computer verständlichen Form. Dies kann entweder durch die Angabe einer Folge von Befehlen, durch die Definition von ineinander verschachtelten Funktionen oder durch die Angabe von Fakten und Regeln geschehen. (weitere Erläuterungen siehe Kapitel 6.2)

6.2 Fachliche Grundlagen zu ausgewählten Themengebieten

Modellierung

Der Modellbegriff ist ein sehr komplexer und vielseitiger Begriff. Wichtig sind dabei immer die folgenden drei Aspekte:

Ein Modell ist ein vereinfachtes Abbild der Wirklichkeit.

Bei der Entwicklung eines Modells werden nur ausgewählte Eigenschaften und Verhaltensweisen berücksichtigt.

Ein Modell dient immer einem bestimmten Zweck.

Dieser Zweck ist für die Auswahl der Eigenschaften von Bedeutung.

Ein Modell ist subjektiv geprägt.

Die Auswahl der Eigenschaften ist subjektiv geprägt.

In der Informatik ist ein Modell ein von Subjekten durch Abstraktion geschaffenes Abbild der realen Welt (Original). Ziel des Modellierens ist es, ein Informatiksystem zu durchdringen und/oder zu erzeugen. Der Einsatz dieses Systems wirkt auf die Realität zurück.

Eine mögliche Klassifizierung von Modellen, die in der Fachwissenschaft Informatik eine Rolle spielen:

- Architekturmodelle (von-Neumann-Rechner, OSI-Referenzmodell, Client-Server-Modell, Farbmodelle...)
- Vorgehensmodelle (Phasenmodell/Wasserfallmodell, objektorientierte Modellierung, ...)
- Entwurfsmodelle (Entity-Relationship-Modell für Datenbanken, Relationenmodell, ...)
- Untersuchungsmodelle (Mathematisches Modell, Simulationsmodell, ...)
- …

(Print: [3])

Für den Informatikunterricht sind dabei von Bedeutung:

- Aufbau eines Computerarbeitsplatzes (Architekturmodell)
- Blockschaltbild der Zentraleinheit (Architekturmodell)
- Struktogramm (Vorgehensmodell)
- Programmablaufplan (Vorgehensmodell)
- Objekt und Klasse (Entwurfsmodell)

Objektorientierung

Die ersten objektorientierten Programmiersprachen wurden bereits in den sechziger und siebziger Jahren des 20. Jahrhunderts entwickelt. Danach setze sich die objektorientierte Herangehensweise bei der Entwicklung großer Softwarepakete immer mehr durch. Diese Handlungsweise beginnt mit der objektorientierten Analyse (OOA) und führt über den objektorientierten Entwurf (OOE) und das objektorientierte Design (OOD) bis hin zur objektorientierten Programmierung (OOP). Von besonderer Bedeutung für die Schule ist dabei, dass die Gestaltung heutiger Benutzeroberflächen unter dem objektorientierten Blickwinkel zu verstehen ist.

Die Begriffe „Klasse“ und „Objekt“ wurden im Kapitel 6.1 bereits definiert, und es muss noch einmal betont werden: Ein Objekt ist immer etwas Konkretes. Es ist über einen Namen identifizierbar und hat stets konkrete Attributwerte. Die Klasse ist der jeweilige zugehörige abstrakte Begriff. Alle Objekte, die aus einer Klasse abgeleitet oder erzeugt werden, haben die gleichen Attribute, aber i. Allg. unterschiedliche Attributwerte und verfügen über die gleichen Methoden (Operationen und Aktionen), die das Verhalten der Objekte dieser Klasse beschreiben.

Wichtige Klassen für die Gestaltung von Benutzeroberflächen sind z. B.:

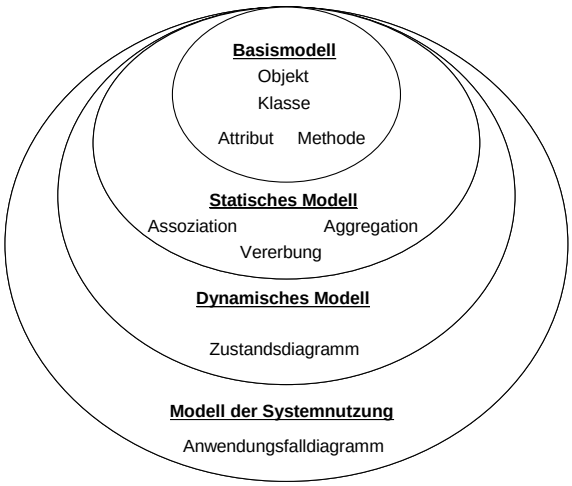
- Fenster (engl. window)
- Beschriftungen (engl. label)
- Ein- oder mehrzeilige Ein-/Ausgabefelder (engl. edit)
- Schaltflächen (engl. button)

Im Unterricht sollte der Begriff „Objekt“ bzw. „Klasse“ zuerst anhand von Beispielen aus dem Alltag erläutert werden. Dabei kann z. B. ein Quartettspiel zu einem bestimmten Thema recht hilfreich sein.

Klassen und Objekte können auch verbal beschrieben werden. Schwieriger wird es schon, die verschiedenen Beziehungen und Abhängigkeiten verbal zu formulieren. Bereits 1997 veröffentlichten G. Booch, J. Rambaugh und I. Jacobson 1997 die erste Version der **U**nified **M**odeling **L**anguage (UML). Sie wurde seit 1994 als Sprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Software entwickelt. Dabei werden folgende Modelle unterschieden

- Basismodell (Klasse, Attribut, Methode, Objekt)
- statisches Modell (Assoziation, Aggregation, Vererbung)
- dynamisches Modell (Zustandsdiagramm)
- Modell der Systemnutzung (Anwendungsfalldiagramm)

Die Beziehungen zwischen diesen Modellen werden in der nachfolgenden Grafik veranschaulicht (Print: [4]):



Im Unterricht an der Oberschule wird nur das Basismodell eine Rolle spielen. Elementare Bestandteile des Basismodells sind die folgenden Symbole zur Darstellung von Klassen und Objekten:

KLASSENNAME	
attribut_1:	Wertebereich
attribut_2:	Wertebereich
...	...
methode_1()	
methode_2()	
...	

objektname: KLASSENNAME	
attribut_1=	Wert
attribut_2=	Wert
...	...
methode_1()	
methode_2()	
...	

Die Erstellung eines Basismodells stellt bereits einen sehr hohen Anspruch an das Abstraktionsvermögen der Schüler. Zwar wird bei der objektorientierten Analyse eines Ausschnitts der realen Welt von Objekten ausgegangen, die in dieser Welt existieren, aber erst durch die Auswahl von „wesentlichen" Eigenschaften und das Ignorieren von „unwesentlichen" Eigenschaften entsteht ein Objekt. Was sind die wesentlichen oder unwesentlichen Eigenschaften? Das kann man ohne Zusatzinformationen nicht entscheiden. Angenommen, es soll eine Klasse „Schüler" definiert und aus dieser Klasse das Objekt „Eva Müller" erzeugt werden. Soll dies zum Beispiel für die Verwaltung einer Bibliothek geschehen, so sind sicher Eigenschaften wie Lesernummer, Geburtsdatum und Schule von Bedeutung. Unwesentlich ist in diesem Zusammenhang die Haarfarbe oder Konfektionsgröße. Für eine Stilberatung beim Friseur hingegen, ist die Haarfarbe eine wesentliche Eigenschaft. Für die Verwaltung der Bibliothek könnte das Basismodell somit folgendermaßen aussehen:

SCHUELER	
schule:	Schule A, Schule B, ...
geschlecht:	w, m
lesernummer:	0001, ..., 9999
lesen()	
ausleihen()	
vorbestellen()	

eva_mueller: SCHUELER	
schule=	Schule B
geschlecht=	w
lesernummer=	0234
lesen()	
ausleihen()	
vorbestellen()	

WICHTIGER HINWEIS: Jedes Objekt ist identifizierbar, d. h. jedes Objekt hat einen Namen, hier z. B. Vor- und Familienname (mittels Unterstrich aneinandergefügt).

Im statischen Modell werden Beziehungen wie

- Aggregation („Abschnitt_1" ist Teil des Dokuments „Text_1")
- Assoziation („Schüler_xyz" besucht „Oberschule_abc")
- Klassenhierarchie („Schüler" ist eine „Person", „Lehrer" ist eine „Person")

dargestellt und reicht damit bereits über das in der Oberschule zu vermittelnde Niveau hinaus. Das gleiche gilt für die verschiedenen Diagrammarten:

- Anwendungsfalldiagramm
- Klassendiagramm
- Aktivitätsdiagramm
- Kollaborationsdiagramm
- Sequenzdiagramm
- Zustandsdiagramm
- Komponentendiagramm
- Einsatzdiagramm

Programmierung mit einer höheren Programmiersprache

Je nach der zu Grunde liegenden Denkweise unterscheidet man folgende Programmierparadigmen:

imperatives Paradigma (Programmierer sagt, was zu tun ist)

deklaratives Paradigma (Programmierer beschreibt die Problemlösung)

Ein Überblick über entsprechende Programmiersprachen gibt die folgende Tabelle:

		Deklarative Sprachen	
	Imperative Sprachen	**Funktionale Sprachen**	**Logische Sprachen**
Programm	Folge von Befehlen	Ineinandergeschachtelte Funktionen	Sammlung von Fakten und Regeln
Kalkül	Variable als Name-Wert-Paar; Wertzuweisung	Rekursion	Backtracking
Beispiele	Fortran, Algol, Pascal, (Logo)	Lisp, Scheme, Logo, ML, Haskel	Prolog

Quer zu diesen drei Paradigmen liegt die objektorientierte Programmierung. Dabei ist die Objektorientierung eigentlich eine Entwurfsmethode, die bei der objektorientierten Analyse beginnt und über den objekt-orientierten Entwurf bis hin zur objektorientierten Programmierung führt. Objektorientierte Sprachen sind SMALTALK, C++, JAVA. Alle objektorientierten Sprachen verfügen mittlerweile über einen umfangreichen Vorrat vordefinierter Klassen, aus denen der Programmierer die entsprechenden Objekte erzeugen kann. Darüber hinaus erlauben sie die Definition von neuen Klassen (Festlegen von Klassennamen, Attributnamen und Methodennamen und Programmierung der Methoden) und die Erzeugung von Objekten dieser Klassen. Objekte können miteinander Botschaften austauschen, die in der Regel die Aufforderung zum Aufruf einer Methode zur Folge haben.

Zur Programmierung nutzerdefinierter Methoden wird zumeist das imperative Paradigma eingesetzt.

Ein erster Schritt in Richtung objektorientierter Programmierung kann die Gestaltung nutzerfreundlicher Oberflächen mit Objekten vordefinierter Klassen sein, z. B.

Programmierung mit ObjectPascal in einer Delphi-Umgebung:

Hier werden die Objekte (Labels, Editfelder, Schaltflächen, ...) aus einer „Komponentenleiste“ mit der Maus in ein Formular eingefügt und anschließend werden die entsprechenden Attributwerte zugewiesen.

Programmierung mit MSWLogo (deutsch):

Mittels entsprechender Grundprozeduren „fensterschaffen“, „knopfschaffen“ u. a. werden die entsprechenden Objekte in einem Fenster erzeugt. Dabei werden die konkreten Attributwerte (z. B. linke obere Ecke, Höhe, Breite.) als aktuelle Parameter der Grundprozeduren angegeben.(Web: [12]))

Programmierung mit JavaScript:

Zur Gestaltung interaktiver HTML-Seiten mittels JavaScript können Nutzereingaben und auch Ausgaben über Formulare realisiert werden. Dabei sind alle mittels entsprechender Tags erzeugten Bestandteile einer HTML-Seite für JavaScript Objekte, so auch die Formularelemente „Ein-Ausgabe-Textfeld“, „Schaltfläche“ u. a:

Nutzerdefinierte Methoden werden meist als FUNKTIONEN oder PROZEDUREN definiert. Funktionen und Prozeduren kann man im Prinzip in allen gängigen Programmiersprachen definieren. Dabei besteht die Definition stets aus dem

Kopf (Schlüsselwort, Name, Liste der formalen Parameter) und dem

Körper (Berechnungs- oder Verarbeitungsvorschrift)

Das Schlüsselwort und die konkrete Syntax sind programmiersprachenabhängig:

LOGO: gleiche Schlüsselworte für Funktionen und Prozeduren: **`def`** bzw. **`lerne`**

PASCAL: Schlüsselworte **`function`** bzw. **`procedure`**

JAVASCRIPT: gleiche Schlüsselworte für Funktionen und Prozeduren: **`function`**

Im Folgenden wird eine Funktion zur Berechnung des Mittelwerts zweier reeller Zahlen in verschiedenen Sprachen definiert (komplette Quelltexte im Anhang):

LOGO:

```
def "mw [[a b] [rg (:a + :b) / 2]
```

oder

```
lerne mw :a :b
 rg (:a + :b) / 2
ende
```

PASCAL:

```
function mw(a, b : real) : real;
begin
 result := (a + b) / 2
end;
```

JAVASCRIPT:

```
mw = new Function("a, b", " var x = parseFloat(a);
var y = parseFloat(b); return (x + y) / 2;");
```

oder

```
function mw(a, b) {
 var x = parseFloat(a);
 var y = parseFloat(b);
 return (x + y) / 2;
}
```

PHP:

```
function mw($a, $b) {
 $x = doubleval($a);
 $y = doubleval($b);
 return ($x + $y) / 2;
}
```

Eine Funktion ist ein Ausdruck (Wert), es wird somit stets ein Funktionswert zurückgegeben (z. B. mittels **output ...**, **result ...** oder **return ..**). Dieser Funktionswert muss anschließend in einer Wertzuweisung oder einer Ausgabeprozedur weiterverarbeitet werden. Beim Aufruf der Funktion müssen die im Kopf angegebenen FORMALEN PARAMETER durch AKTUELLE PARAMETER ersetzt werden:

LOGO:

```
dzl mw 5 6
komtaboxtext "eErgebnis mw 5 6
```

PASCAL:

```
ErgebnisieheText := FloatToStr(mw(5,6));
```

JAVASCRIPT:

```
eErgebnisiehevalue = mw(5, 6);
document.write(mw(5, 6));
window.alert(mw(5, 6));
```

PHP:

```
echo mw(5, 6);
```

Man kann die gleiche Berechnungsvorschrift als Prozedur formulieren. Allerdings ist dies i. Allg. nicht sehr sinnvoll, da dann kein Wert für weitere Rechnungen zur Verfügung steht:

LOGO:

```
def "mwp [[a b] [dzl (:a + :b) / 2]
```

oder

```
lerne mwp :a :b
 dzl (:a + :b) / 2
ende
```

JAVASCRIPT:

```
mwp = new Function("a, b", "var x = parseFloat(a);
           var y = parseFloat(b); window.alert ((:x + :y) / 2");
```

oder

```
function mwp(a, b) {
 var x = parseFloat(a);
 var y = parseFloat(b);
 window.alert ((:x + :y) / 2)
}
```

PHP:

```
function mwp($a, $b) {
 $x = doubleval($a);
 $y = doubleval($b);
 echo ($x + $y) / 2;
}
```

Eine Prozedur wird mit ihrem Namen aufgerufen. Auch dabei müssen die formalen Parameter durch aktuelle Parameter ersetzt werden:

LOGO:

```
mwp 5 6
```

JAVASCRIPT:

```
mwp(5, 6)
```

PHP:

```
mwp(5, 6)
```

Programmierung mit anderen Programmierumgebungen

Neben der Programmierung mit „echten“ Programmiersprachen können die algorithmischen Grundstrukturen der Programmierung auch an didaktisch reduzierten Programmierumgebungen (Print: [8]) vermittelt werden. Besonders für die Oberschule als geeignet haben sich die Umgebung „Scratch“ und „Robot Karol“ erwiesen, die als freie Versionen zur Verfügung stehen. Im Folgenden sollen diese Umgebungen etwas näher dargestellt werden:

Umgebung	**Robot Karol**	**Scratch**
Autoren	Freiberger, Krsko	Lifelong-Kindergarten-Gruppe am Media-Lab des MIT
Webquelle	**(Web: [8])**	**(Web: [17])**
Systemanforderungen	Installation unter Windows	betriebssystemunabhängiges Java-Programm, benötigt Java Laufzeitumgebung auf dem Rechner (JRE)
Programmierprinzip	Steuern eines Roboters in einer Bildschirmwelt mittels einer eigenen Programmiersprache **(Web: [14])**	Steuern einer Katze in einer Bildschirmwelt mittels grafischer Programmierung.
Klassen	Welt, Karol, Quader, Ziegel, Marke	Bühne (z.B. Hintergrund), Figuren (z.B. Katze, Ball, Button)

Beschreibung ausgewählter Klassen

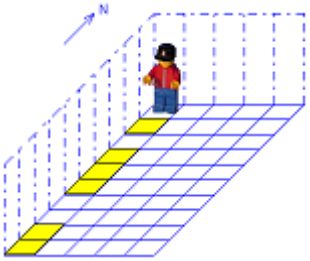
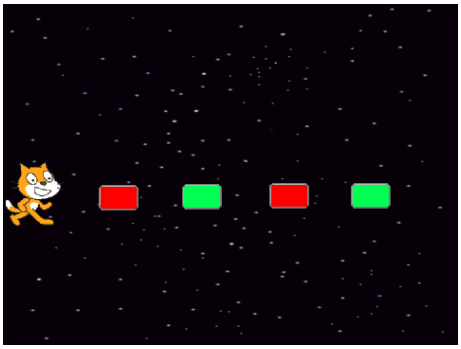
KAROLS_WELT	
breite:	1 ... 100
laenge:	1 ... 100
hoehe:	1 ... 10
dateiname:	*.kdw
neue_welt()	
welt_loeschen()	
welt_wiederherstellen()	

BUEHNE	
hintergrund:	<name>
breite:	1 ... 480 px
hoehe:	1 ... 360 px
dateiname:	*.sb
hintergrund_malen	
hintergrund_importieren	
hintergrund_fotografieren	

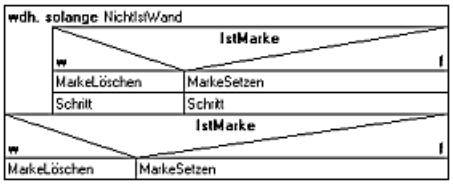
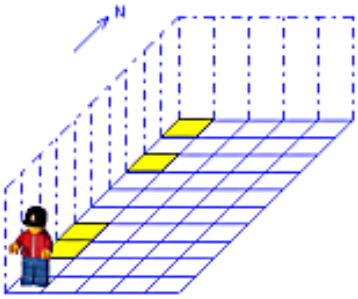
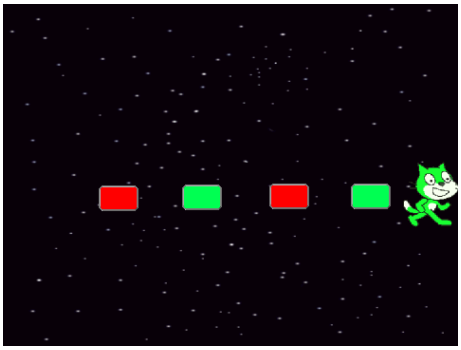
KAROL	
blickrichtung:	Nord, Süd, Ost, West
sprunghoehe:	1 ... 10
tragen_max:	1 ... 999
tragen_start:	0 ... 999
schritt()	
linksdrehen()	
rechtsdrehen()	
hinlegen()	
aufheben()	
markesetzen()	
markeloeschen()	
warten()	
ton()	
langsam()	
schnell()	
beenden()	
...	

FIGUR	
name:	<name>
kostüm:	<kostümname>
position(x):	x
position(y):	y
drehmodus:	frei drehbar, links/rechts, nicht drehbar
richtung:	-179° ... 180°
gehe_n-er-schritt	
drehe_dich_um_n-grad	
pralle_vom_rand_ab	
sage_“xyz“	
zeige_dich	
sende_x_an_alle	
hinterlasse_abdruck	
...	

Neben den umfangreichen Methoden zum Erkennen seiner Umwelt können für Karol auch eigene Methoden definiert werden.

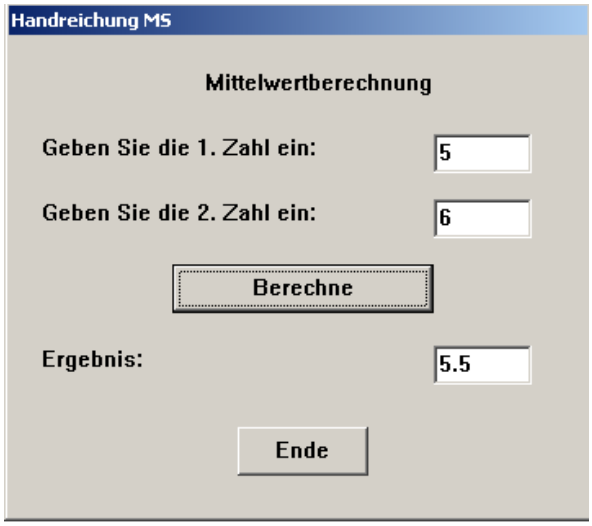
Umgebung	**Robot Karol**	**Scratch**
Erstellen einer Welt	in einer grafische Oberfläche, Karol muss an die Stelle gesteuert werden, wo in der Welt etwas verändert werden soll	in einer grafischen Oberfläche
Programmieren einer Problemlösung	separates Programmfenster, Texteingabe der Befehle mit Schlüsselworterkennung und Hilfe per rechter Maustaste	
Beispielaufgabe	Karol steht in einer beliebigen Welt hinten links mit Blickrichtung Süd und soll das Muster in der Reihe vor sich invertieren.	Katze soll bis zum Rand gerade aus laufen. Auf ihrem Weg soll sie ihre Farbe, je nach Buttonfarbe, ändern (rot→grün→rot→grün)
Startwelt		

Lösung	`wiederhole solange NichtIstWand`
	`  wenn IstMarke dann`
	`  MarkeLöschen`
	`  Schritt`
	`  sonst`
	`  MarkeSetzen`
	`  Schritt`
	`  *wenn`
	`*wiederhole`
	`wenn IstMarke dann`
	`  MarkeLöschen`
	`sonst`
	`  MarkeSetzen`
	`*wenn`
		Skript Katze:
		 

		
Ergebniswelt		

Anhang:

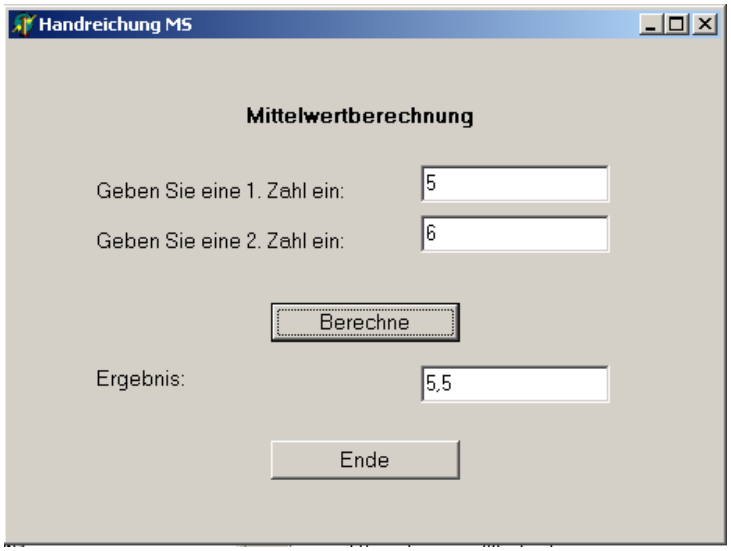
LOGO (MSW LOGO, DEUTSCH) (Web: [12]):




```
lerne beenden
        ;Prozedur zum Löschen des Fensters
        fensterlöschen "fmw
ende

lerne mw :a :b
        ;Funktion zur Berechnung des Mittelwerts
        rg (:a + :b) / 2
ende

lerne mweinaus
        ;Prozedur zur Eingabe, Berechnung und Ausgabe
        setze "x erstes komtaboxtextliste "eZahl1
        setze "y erstes komtaboxtextliste "eZahl2
        komtaboxtext "eErgebnis mw :x :y
ende

lerne start
        ;Startprozedur zur Erzeuguzng der Oberfläche
        fensterschaffen "hauptfenster "fmw [Handreichung MS]           10  10 180 160 []
        statuszeileschaffen "fmw "lTitel [Mittelwertberechnung]        60  10 100  10
        statuszeileschaffen "fmw "lZahl1 [Geben Sie die 1. Zahl ein:]  10  30 110  10
        komtaboxschaffen    "fmw "eZahl1                              130  30  30  10
        statuszeileschaffen "fmw "lZahl2 [Geben Sie die 2. Zahl ein:]  10  50 110  10
        komtaboxschaffen    "fmw "eZahl2                              130  50  30  10
        knopfschaffen       "fmw "bBerechne [Berechne]                 50  70  80  15~
        [mweinaus]
        statuszeileschaffen "fmw "lErgebnis [Ergebnis:]                10  95 110  10
        komtaboxschaffen    "fmw "eErgebnis                           130  95  30  10
        knopfschaffen       "fmw "bEnde [Ende]                         70 120  40  15~
        [beenden]
ende
```

Bemerkungen: In einem Logoquelltext können zwischen den Attributwerten beliebig viele Leerzeichen stehen, um z. B. die Positionierung und Größe der Objekte besser lesbar zu machen (siehe Zeile: **`komtaboxschaffen`** ...). Allerdings müssen alle aktuellen Parameter (Attributwerte) auf einer Zeile stehen. Sollte dieses nicht möglich sein, kann durch das Einfügen der Zeichens „~“ (Tilde) der Zeilenumbruch für den Interpreter aufgehoben werden (siehe Zeile: **`knopfschaffen`** ...).

PASCAL (MIT DELPHI):



Der Quelltext der Project-Datei wird automatisch erzeugt.

Quelltext der Unit:

```
unit mittelwert;
interface
uses
        Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
        StdCtrls;
type
        TFormular1 = class(TForm)
                lTitel: TLabel;
                lZahl1: TLabel;
                lZahl2: TLabel;
                eZahl1: TEdit;
                eZahl2: TEdit;
                bBerechne: TButton;
                lErgebnis: TLabel;
                eErgebnis: TEdit;
                bEnde: TButton;
                procedure bBerechneClick(Sender: TObject);
                procedure bEndeClick(Sender: TObject);
                private
                {Private-Deklarationen}
        public
                {Public-Deklarationen}
        end;
var
        Formular1: TFormular1;
implementation
{$R *.DFM}
function mw(a, b : real) : real;
// Funktion zur Mittelwertberechnung
begin
        result := (a + b) / 2
end;

procedure TFormular1.bBerechneClick(Sender: TObject);
// Prozedur zur Eingabe, Berechnung und Ausgabe
var x, y : real;
begin
        x := StrToFloat(eZahl1.Text);
        y := StrToFloat(eZahl2.Text);
        eErgebnisieheText := FloatToStr(mw(x,y));
end;
procedure TFormular1.bEndeClick(Sender: TObject);
// Prozedur zum Beenden
begin
        close
end;
end.
```

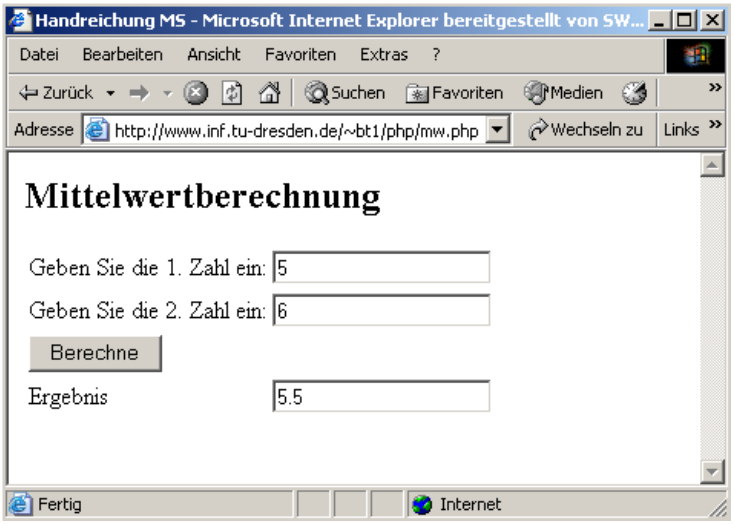
JAVASCRIPT (Web: [15], [16]):

Bemerkung: JavaScript wird vom Browser des Client interpretiert. Somit können HTML-Dokumente mit JavaScript-Code lokal in einem JavaScript-fähigen Browser getestet werden.


```
<!doctype html public "-//W3C//DTD HTML 4.0 //EN">
<html>
<head>
<title>Handreichung MS</title>
<meta name="author" content="bt1">
<meta name="generator" content="Ulli Meybohms HTML EDITOR">
<script type = "text/javascript">
<!--
function mw(a, b) {
// Funktion zur Berechnung des Mittelwerts
        var x = parseFloat(a);
        var y = parseFloat(b);
        return (x + y) / 2;
}
//-->
</script>
<noscript>Bitte JavaScript aktivieren!</noscript>
</head>
<body>
<h2>Mittelwertberechnung</h2>
<form>
        <table>
                <tr>
                        <td>Geben Sie die 1. Zahl ein:</td>
                        <td><input type = "text" name = "eZahl1" value = ""></input></td>
                </tr>
                <tr>
                        <td>Geben Sie die 2. Zahl ein:</td>
                        <td><input type = "text" name = "eZahl2" value = ""></input></td>
                </tr>
                <tr>
                        <td><input type = "button" name = "bBerechne" value = "Berechne"
                        onclick = "eErgebnisiehevalue = mw(eZahl1.value,
                eZahl2.value)"></td>
                </tr>
                <tr>
                        <td>Ergebnis</td>
                        <td><input type = "text" name = "eErgebnis" value = ""</td>
                </tr>
        </table>
</form>
</body>
</html>
```

PHP (Web: [17]):

Bemerkung: php wird stets vom Server interpretiert. Somit müssen php-Dateien zum Testen auf einen php-fähigen Webserver hochgeladen werden. Jedoch kann man sich zu Testzwecken auch auf einem nicht im Netz eingebundenen Rechner lokal einen Webserver installieren.




```
<!doctype html public "-//W3C//DTD HTML 4.0 //EN">
<html>
<head>
<title>Handreichung MS</title>
<meta name="author" content="bt1">
<meta name="generator" content="Ulli Meybohms HTML EDITOR">
<?php
function mw($a, $b) {
// Funktion zur Berechnung des Mittelwerts
        $x = doubleval($a);
        $y = doubleval($b);
        return ($x + $y) / 2;
}
?>
</head>
<body>
<h2>Mittelwertberechnung</h2>
<form action="mw.php" method="post">
        <table>
                <tr>
                        <td>Geben Sie die 1. Zahl ein:</td>
                        <td><input type = "text" name = "eZahl1"
                        value = "<?php if (isset ($bBerechne)) echo $eZahl1;?>">
                        </input></td>
                </tr>
                <tr>
                        <td>Geben Sie die 2. Zahl ein:</td>
                        <td><input type = "text" name = "eZahl2"
                        value = "<?php if (isset ($bBerechne)) echo $eZahl2; ?>">
                        </input></td>
                </tr>
                <tr>
                        <td><input type = "submit" name = "bBerechne" value =
                        "Berechne"></td>
                </tr>
                <tr>
                        <td>Ergebnis</td>
                        <td><input type = "text" name = "eErgebnis"
                        value = "<?php if (isset ($bBerechne)) echo mw($eZahl1, $eZahl2);
                        ?>"</td>
                </tr>
        </table>
</form>
</body>
</html>
```

7. Quellen und weiterführende Literatur

7.1 Printquellen

[1] Norbert Breier, Steffen Friedrich: *Informatische Grundbildung* Paetec Verlag 2003 ISBN 3-89818-603-2

[2] Lutz Engelmann: *Duden Angewandte Informatik* Paetec Verlag 2001 ISBN 3-89818-035-2

[3] Marco Thomas: *Informatische Modelle zur Strukturierung von Anfangsunterricht*, in: P. Hubwieser, Informatische Fachkonzepte im Unterricht, GI-Edition Lecture Notes in Informatics, ISBN 3-88579-361-X

[4] Peter Forbrig, *Objektorientierte Softwareentwicklung mit UML*, Fachbuchverlag Leipzig im Carl Hanser Verlag München, 2001, ISBN3-446-21572-7

[5] Volker Claus, Andreas Schwill: *Schülerduden Informatik*, Dudenverlag, Mannheim 1997, ISBN 3-411-04483-7

[6] Volker Claus, Andreas Schwill: *Duden Informatik*, Dudenverlag, Mannheim u.a., 1995, ISBN 3-411-05232-5

[7] Joachim Ziegenbalg: *Algorihmen von Hamurapi bis Gödel*, Spektrum Akademischer Verlag, Heidelberg, 1996, ISBN 3-8274-0114 – 3

[8] Raimond Reichert, Jörg Nievergelt, Werner Hartmann: *Programmieren mit Kara*; Springer 2004, ISBN 3-540-40362-0

7.2 Weiterführende Literatur

Werner Hartmann, Michael Näf, Peter Schäuble: *Informationsbeschaffung im Internet Grundlegende Konzepte verstehen und umsetzen*, Orell Füssli Verlag 2000, ISBN 3-280-02793-4

Michael Näf, Patrick Streule, Werner Hartmann: *Risiko Internet? Sicherheitsaspekte bei der Internet-Benutzung*, Orell Füssli Verlag 2000, ISBN 3-280-02770-5

Andreas Holzinger: *Basiswissen IT / Informatik. Bd.1-3*, Vogel-Buchverlag, 2002 ISBN 3-8023-1897-8

Volker Ulm: *Objekte in Grafiken*, Uni Bayreuth 2003, ISBN 3-00-010566-22

Karsten Mielke, Ralf Sagorny: *Programmentwicklung mit UML*, Bildungsverlag EINS 2003, ISBN 3-427-01145-3

Peter Rechenberg, *Was ist Informatik?*, Carl Hanser Verlag München, 1991, ISBN 3-446-16324-7

Hans-Jürgen Appelrath, Dietrich Boles, Volker Claus, Ingo Wegener, *Starthilfe Informatik*, B. G. Teubner Verlag, Stuttgart 1998, ISBN 3-519-00241-8

7.3 Webquellen (aktualisiert 04/2018)

[1] https://www.schule.sachsen.de/lpdb/web/downloads/lp_ms_informatik_2009.pdf?v2
Lehrplan Informatik Oberschule

[2] https://www.schule.sachsen.de/lpdb/web/downloads/895_Lehrplanmodell.pdf?v2
Dokument zum Lehrplanmodell

[3] https://www.schule.sachsen.de/lpdb/web/downloads/lp_ms_technik_computer_2009.pdf?v2
Lehrplan Informatik Oberschule

[4] https://www.schule.sachsen.de/lpdb/
Lehrpläne aller Fächer

[5] https://www.schule.sachsen.de/lpdb/web/downloads/1566_Fachuebergreifender_und_faecherverbindender_Unterricht.pdf?v2
Grundsatzpapier: Fachübergreifender und fächerverbindender Unterricht

[6] https://www.sachsen.schule/~knapp/index.php?auswahl=caesar
Umsetzung der Cäsar-Verschlüsselung ohne Schlüsselwort mit Tabellenkalkulation

[7] https://cms.sachsen.schule/scratch/grundlagen/
Materialien und Links zur Programmierung mit Scratch für die Oberschule

[8] https://www.mebis.bayern.de/infoportal/faecher/mint/inf/robot-karol/
Materialien und Links zur Programmierung mit Robot Karol

[9] https://www.sachsen.schule/mindjet/bestellung.php
MindManager Smart kostenlos für Schulen

[10] http://userpage.fu-berlin.de/~medienfo/hp/Spiele/Homepage_Computerspiele.html
Computerspiele – Wirkungen von Medien auf Kinder und Jugendliche

[11] http://www.uml.org
UML™ Resource Page

[12] https://www.ph-ludwigsburg.de/logo.html
Quelle für deutsche Version des freien MSW Logo:

[13] https://www.mebis.bayern.de/infoportal/faecher/mint/inf/robot-karol/
Mini-Language

[14] https://selfhtml.org/
Stefan Münz – Selfhtml

[15] https://www.phase5.info/
HTML-Editor: Phase 5

[16] http://www.selfphp.de/
Selfphp

[17] https://scratch.mit.edu
Lifelong-Kindergarten-Gruppe am Media-Lab des MIT

https://www3.sachsen.schule/sbs/lehrenlernen/mittelschule/naturwissenschaften/informatik/
Informatik-Wegweiser, Sammlung von Quicklinks und anderen Materialien auf dem SBS

https://www3.sachsen.schule/sbs/lehrenlernen/
Sammlung von Links und anderen Materialien auf dem SBS

https://www3.sachsen.schule/sbs/lehrenlernen/mittelschule/naturwissenschaften/informatik/
Startseite zum Fach Informatik an Oberschulen auf PäPIKK

http://www.log-in-verlag.de/index.html
Webseite der LOG IN