Landesinstitut für Schule und Weiterbildung

Gymnasiale Oberstufe

Informatik

Lehrplanentwurf

(Stand: 15. August 1998)

Inhalt

Seite

1. Aufgaben und Ziele des Faches 5

1.1 Didaktische Konzeption und fachliche Anforderungen 5

1.2 Zusammenarbeit mit anderen Fächern 8

2. Bereiche, Themen, Gegenstände 10

2.1 Bereiche: Herleitung und didaktische Funktion 10

2.2 Zuordnung der Themen und Gegenstände zu den Bereichen 10

2.2.1 Fachliche Inhalte 10

2.2.1.1 Modellieren und Konstruieren 12

2.2.1.2 Analysieren und Bewerten 14

2.2.2 Lernen im Kontext 16

2.2.3 Methoden und Formen selbständigen Arbeitens 20

2.2.3.1 Fachspezifische Vorgehensweisen 20

2.2.3.2 Selbstständige und projektorientierte Arbeitsformen 21

2.2.3.3 Fachübergreifende und fächerverbindende Sichtweisen 22

2.3 Obligatorik und Freiraum 22

2.3.1 Stufung der Obligatorik 23

2.3.1.1 Jahrgangsstufe 11 24

2.3.1.2 Jahrgangsstufen 12 und 13 24

2.3.1.3 Zusammenarbeit mit einer Naturwissenschaft 25

2.3.2 Zur Konstruktion möglicher Unterrichtssequenzen 25

2.3.2.1 Modularisieren und Entwickeln einer geeigneten Organisation 25

2.3.2.2 Aufdecken der Funktion bekannter und Entwickeln neuer Werkzeuge 26

2.3.2.3 Fortschreitende Analysetechnik und Bewertung 26

2.3.3 Darstellung der Obligatorik am Beispiel unterschiedlicher Sprachkonzepte 27

2.3.3.1 Imperativer Ansatz 27

2.3.3.2 Objektorientierter Ansatz 29

2.3.3.3 Wissensbasierter Ansatz bei Verwendung einer PROLOG-Programmierumgebung 31

2.3.3.4 Funktionaler Ansatz 32

2.3.4 Zusammenfassung inhaltlicher und methodischer Kompetenzen 34

3. Unterrichtsgestaltung/Lernorganisation 36

3.1 Grundsätze der Unterrichtsgestaltung 36

3.2 Gestaltung der Lernprozesse 37

3.2.1 Kriterien für die Auswahl von Unterrichtsinhalten 38

3.2.2 Lern- und Arbeitsorganisation im Fach 39

3.2.2.1 Konventionelle Arbeitsformen im Fach Informatik 39

3.2.2.2 Facharbeit 40

3.2.3 Fachübergreifende, fächerverbindende und projektorientierte Lern- und Arbeitsorganisation 41

3.2.4 Besondere Lern- und Arbeitsformen 42

3.3 Grund- und Leistungskurse 43

3.4 Sequenzbildung **45**
3.4.1 Sequenzen für die Jahrgangsstufe 11 45
3.4.2 Sequenzen für die Jahrgangsstufen 12 und 13 57
3.4.2.1 Vertiefung eines paradigmatischen Ansatzes 58
3.4.2.2 Paradigmenunabhängige Sequenzen für die Jahrgangsstufen 12 und 13 66
3.5 Mädchen und Jungen im mathematisch-naturwissenschaftlich-technischen Unterricht **71**
4. Lernerfolgsüberprüfungen **72**
4.1 Grundsätze **72**
4.2 Beurteilungsbereich „Klausuren“ **72**
4.2.1 Allgemeine Hinweise 72
4.2.2 Fachspezifische Hinweise zur Aufgabenstellung, Korrektur und Bewertung von Klausuren/Facharbeiten 73
4.2.2.1 Grundsätze zur Konstruktion von Aufgabenarten 73
4.2.2.2 Planung von Klausuren 73
4.2.2.3 Korrektur, Bewertung und Rückgabe von Klausuren 75
4.2.2.4 Korrektur und Bewertung von Facharbeiten 76
4.3 Beurteilungsbereich „sonstige Mitarbeit“ **76**
4.3.1 Allgemeine Hinweise 76
4.3.2 Anforderungen und Kriterien zur Beurteilung der Leistungen im Beurteilungsbereich „sonstige Mitarbeit“ 77
5. Die Abiturprüfung **82**
5.1 Allgemeine Hinweise **82**
5.2 Beschreibung der Anforderungsbereiche **82**
5.3 Die schriftliche Abiturprüfung **85**
5.3.1 Aufgabenarten der schriftlichen Abiturprüfung 85
5.3.1.1 Beispiele für typische Aufgabenarten 85
5.3.1.2 Grundsätze zur Konstruktion von Prüfungsaufgaben 86
5.3.2 Einreichen von Prüfungsvorschlägen 86
5.3.3 Bewertung der schriftlichen Prüfungsleistungen 87
5.3.4 Beispiele für Prüfungsaufgaben in der schriftlichen Abiturprüfung 87
5.4 Die mündliche Abiturprüfung **98**
5.4.1 Aufgabenstellung für den ersten Teil der mündlichen Prüfung 98
5.4.2 Aufgabenstellung für den zweiten Teil der mündlichen Prüfung 99
5.4.3 Bewertung der Prüfungsleistungen 99
5.4.4 Beispiel für Prüfungsaufgaben in der mündlichen Prüfung 99
5.5 Die besondere Lernleistung **100**
6. Hinweise zur Arbeit mit dem Lehrplan **101**

1. Aufgaben und Ziele des Faches

1.1 Didaktische Konzeption und fachliche Anforderungen

Im Allgemeinen werden mit *Informatik* zuerst Computer als sichtbare Resultate dieser Wissenschaft und der Informations- und Kommunikationstechnologien assoziiert. In Alltag, Beruf und Schule wird der PC immer stärker als *Werk*-Zeug genutzt. Ständig erweitert sich das Spektrum beruflicher und wissenschaftlicher Anwendungsfelder, in denen ausgeklügelte und effiziente „Computer-Lösungen“ in weltumspannenden Netzen zur Verfügung stehen.

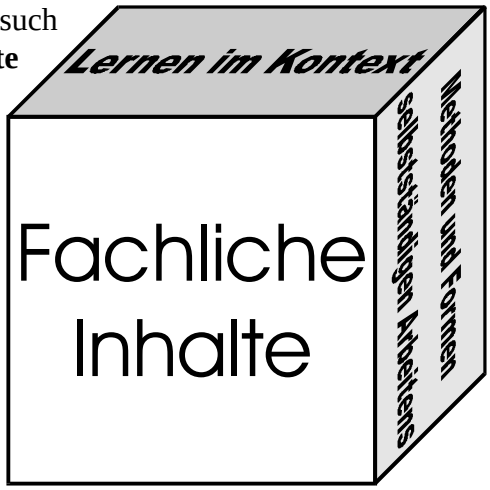
Während die Technikwissenschaften die materiellen Grundlagen bereitstellen, fällt der Informatik die Aufgabe der systematischen Entwicklung von Verfahren, Modellen und übergeordneten Prinzipien effizienter Informationsbearbeitung zu.

In dieser Funktion liefert sie das *Denk*-Zeug, mit dessen Hilfe sich Informatiksysteme entwikkeln, gestalten und zielgerichtet anwenden lassen: die gewünschten Wirkungen sind Beiträge zur Lösung möglichst vielfältiger und komplexer Problemstellungen der menschlichen Lebenswelt. Die unvermeidlichen Nebenwirkungen zeigen sich in teilweise umwälzenden Veränderungen der Alltagswelt und führen ihrerseits zu ständig neuen Herausforderungen.

In der Schule ist die zunehmende Nutzung moderner Informations- und Kommunikationssysteme innerhalb des Unterrichts der verschiedenen Fächer Ausdruck eines gewollten medialen Wandels.

Das Schulfach Informatik in der gymnasialen Oberstufe ist nicht durch den bloßen *Einsatz* von Informatiksystemen *im Anwendungsbereich* definiert und legitimiert, sondern es bleibt unter wissenschaftspropädeutischen Aspekten vornehmlich *der Arbeit an Modellen, Strategien und Techniken bis hin zur praktischen Umsetzung* verpflichtet.

Auch wenn man die Studierfähigkeit der Schülerinnen und Schüler als ein essentielles Ziel des Oberstufenunterrichts ansieht, kann sich der schulisch aufzuarbeitende Teil der Informatik nicht allein der Systematik der universitären Bezugsdisziplin unterwerfen. Bei der rasanten Eigendynamik dieses Leitfaches wird jeder Versuch fehlschlagen, die zentralen **fachlichen Inhalte** überwiegend mit Anleihen aus dem Wissenschaftsbereich zu beschreiben, dabei eine schulgeeignete, das Wesen des Faches vermittelnde Auswahl zu treffen und diese im Spannungsfeld konkurrierender didaktischer Ansätze in ein plausibles Unterrichtskonzept umzusetzen.



Erfolgversprechender erscheint eine *schulspezifische* Auswahl, die einem allgemeineren Bildungsanspruch folgend dem **Lernen im Kontext** Raum gibt, dabei die Anwendungen zum Ausgangspunkt analysierender Betrachtung und die Auswirkungen zur Grundlage kritischer Reflexion macht. Verbunden mit geeigneten unterrichtlichen **Methoden und Formen selbständigen Arbeitens** lässt sich die

Schulinformatik damit auf drei Ebenen eines **„didaktischen Würfels“** projizieren, die im Folgenden weiter zu präzisieren sind.

Als Reaktion auf die umfassende Nutzung von Computern und der modernen Verfahren zur Informationsbearbeitung hat die Schule die Aufgabe übernommen, die zielgerichtete Anwendung von Computern als aktuelle Kulturtechnik in den Unterrichtsfächern zu verankern und medienpädagogisch reflektiert zu vermitteln. Eingelöst wird dies schwerpunktmäßig in der informations- und kommunikationstechnologischen Grundbildung der Sekundarstufe I.

Ebenso wie sich die Informatik mit den von ihr entwickelten allgemein gültigen Techniken und Prinzipien als theoretischer Horizont weit über das Feld der konkreten Anwendungen spannt, liegt die unterrichtliche Arbeit an solchen Prinzipien und Modellen jenseits der direkten Nutzung und geht damit qualitativ weit über den Blickwinkel einer Grundbildung hinaus.

Ins Zentrum unverzichtbarer **fachlicher Inhalte** des Informatikunterrichts in der gymnasialen Oberstufe rücken generalisierbare Techniken zur Modellbildung und zur (Weiter)Entwicklung von Anwendungssystemen sowie Verfahren zur Analyse und Bewertung vorliegender Informatiksysteme. Diese Position sucht unter wissenschaftspropädeutischem Aspekt die Nähe zur Softwaretechnologie als einer wesentlichen Ausprägung des Hochschulfaches Informatik.

Der konkrete Unterricht hat dabei drei anspruchsvolle Themenfelder im Zusammenhang mit der fortschreitenden Entwicklung von Informatiksystemen zu erschließen: Abschnitte, die auf das im Fach angelegte *Strukturwissen* ausgerichtet sind, lenken den Blick sowohl auf die Grundlagen als auch auf Grenzen fortschreitend technisierbarer Wissensbearbeitung. Fragen nach den *Wirkprinzipien* von Informatiksystemen sollen klären, wie diese aufgebaut sind, nach welchen Funktionsprinzipien ihre Komponenten effizient zusammenwirken und wie diese sich in größere Systemzusammenhänge einordnen lassen. *Analyse und Bewertung* vorliegender Entwürfe bilden schließlich einen dritten Komplex: Wie werden durch die Entwicklung, Gestaltung und Anwendung von Informatiksystemen Probleme der Lebenswelt gelöst, in welcher Weise werden relevante Veränderungen und damit u.U. neue Probleme geschaffen und welche Verantwortungen erwachsen daraus?

Für die Informatik ergibt sich innerhalb des mathematisch-naturwissenschaftlichen Aufgabenfeldes ein besonders weiter Spannungsbogen fachlicher Arbeit: Die praxisnahe, informatikspezifische Problemanalyse geht einer fachspezifischen Modellbildung voraus; sie erlaubt die Entwicklung von Lösungsstrategien, die bei geeigneter Implementation zu arbeitsfähigen „Computer-Lösungen“ führen. Jeder einzelne Schritt und insbesondere die an die Praxis rückgekoppelte Bewertung der „Lösung“ kann auf die voraufgegangenen Arbeitsabschnitte zurückwirken. Dabei gehören die analytische Arbeit und das konstruktiv-synthetische Vorgehen untrennbar zusammen. Sie bleiben beide eingebettet in einen rückgekoppelten zyklischen Prozess, der die Problemsicht und die Ausgangssituation durch die gefundene Lösung bzw. Lösungsbewertung verändern kann.

Verbunden mit informatikspezifischen unterrichtlichen **Methoden** fördert der oben beschriebene Entwicklungsprozess gleichzeitig **Formen selbständigen Arbeitens** und eine Reihe sehr viel allgemeinerer Qualifikationen und Bildungsziele:

Schon die Problemanalyse verlangt die Fähigkeit, komplexe Zusammenhänge mit gedanklicher Schärfe zu durchdringen, Strukturen und Wirkprinzipien herauszufiltern und sie im Abgleich mit anderen Schülerinnen und Schülern sprachlich korrekt zu artikulieren.

Die sich anschließende zielgerichtete Modellbildung wird zur Brücke in die „computerisierbare Welt“. Die für die Informatik typischen Modellierungstechniken erhalten bei den Schülerinnen und Schülern u.a. dadurch Anreiz und Motivation, dass sich mit ihnen Lösungsstrategien entwerfen und formulieren lassen, die - auf die Maschine umgesetzt - zum erfahr- und überprüfbaren Resultat eines handlungs- und produktorientierten Unterrichts werden.

Während die Strategieentwicklung neben gedanklicher Stringenz insbesondere die Phantasie, die Kreativität und allgemein die geistige Beweglichkeit fördert, ist die sich anschließende Implementationsphase eher ein gutes Übungsfeld für den sicheren Umgang mit formalsprachlichen Symbolen und Notationen.

Im Rahmen projektorientierten Unterrichts zur (Weiter)Entwicklung von Softwaresystemen steigender Komplexität werden die Teamfähigkeit in der einzelnen Gruppe und das Zusammenwirken der beteiligten Teams geschult, wodurch die Kommunikations- und Kooperationsfähigkeit und die Bereitschaft dazu gefördert werden.

Um eine durch Überlegung gefundene Problemlösung beurteilen zu können, muss man sie ggf. unter vereinfachten Bedingungen realisieren und ausprobieren: Produkt- und Handlungsorientierung kommen in der Phase der konkreten Anwendung noch einmal besonders zum Tragen.

Die sich anschließende Bewertung wird nicht nur das fertige System im Rahmen der Modellannahmen einer möglichst vielseitigen Beleuchtung aussetzen, sondern auch den voraufgegangenen Entwicklungsprozess und die erkennbaren Folgen für den Anwendungsbereich mit in die Reflexion einbeziehen.

Ansätze zur Einschätzung der Möglichkeiten und Grenzen des Computereinsatzes und die Thematisierung seiner gesellschaftlichen Auswirkungen regen auf der Grundlage eines soliden Fachwissens die Urteils- und Kritikfähigkeit an.

Ein auf differenzierter Gruppenarbeit basierter handlungs- und produktorientierter Unterricht, der an praxisnahen Informatik-Projekten vielfältige fachspezifische, allgemein bildende und soziale Kompetenzen vermittelt, ist als lohnendes Ziel zu verstehen, auf das in kleinen und gesicherten Schritten hingearbeitet werden muss. Dabei steht den vielfältigen Chancen, die gerade aus dem Informatikunterricht für eine Allgemeinbildung moderner Ausprägung erwachsen, die verfügbare Unterrichtszeit begrenzend entgegen.

Um einer Überfrachtung mit immer neuen Inhalten und allgemeinen Anforderungen zu begegnen, muss der Katalog inhaltlicher Themen eher schlank gestaltet werden. Nur so lässt sich der nötige Freiraum gewinnen, um den Unterricht in Form und Methoden stärker auf jeweils aktuelle Anforderungen hin zu orientieren.

1.2 Zusammenarbeit mit anderen Fächern

Die in Jahrhunderten gewachsene Differenzierung wissenschaftlicher Disziplinen hat in der Schule zur entsprechenden Aufteilung in die wissenschaftsorientierten Unterrichtsfächer geführt. Während die Hochschule von einigen Vorteilen dieser Entwicklung profitieren kann, muss die Schule auf eine im „Schülerkopf“ integrierbare „Sicht der Dinge“ hinarbeiten, wobei die unterrichtliche Bearbeitung inter- statt intradisziplinärer Themenschwerpunkte helfen kann.

Als Ausgangspunkt fachübergreifenden Lernens eignen sich aus der Sicht der Informatik insbesondere solche Problemkreise, bei denen von Anwendungsfällen verschiedenster Fachgebiete abstrahiert werden kann, sodass sich idealtypisch informatische Strukturen herausarbeiten lassen.

Die gewünschte Kooperation gelingt z. B. mit der Mathematik, indem der gemeinsam verwendete Algorithmenbegriff thematisiert und seine Implikationen für die „Berechenbarkeit“ erschlossen werden.

Die sprachlichen Fächer bewegen sich bei Fragen nach Syntax, Semantik, Erkennen und Verstehen von Texten in Themenfeldern, in denen die Informationsverarbeitung aktuelle Schwerpunkte setzt.

Die Fächer des gesellschaftswissenschaftlichen Aufgabenfeldes können bei Themen wie „Datenschutz“, „Computerkriminalität“, „Auswirkungen der Informations- und Kommunikationstechnologien“ usw. mit der Informatik kooperieren.

Alle Fächer des mathematisch-naturwissenschaftlich-technischen Aufgabenfeldes, in denen die Modellierung und Simulation komplexer Wirkungsgefüge eine besondere Rolle spielen, bieten sich für eine Kooperation mit der Informatik an.

Mit der Analyse „computerisierbarer“ Prozesse unserer Lebenswelt lässt sich schließlich die Wechselbeziehung zwischen Mensch und Maschine und damit die Abgrenzung zwischen individuellem Handeln und maschinellem Funktionieren in den Mittelpunkt rücken. Hier bewegt man sich im Grenzbereich zwischen Informatik und praktischer Philosophie.

Neben der vielfältig erweiterbaren Palette fächerübergreifender Themenbereiche lassen sich auch die Intensität und die Organisationsformen der gewünschten Fächerkooperation variieren.

Seit jeher greift der Informatikunterricht bei der Arbeit in Anwendungsbereichen auf Inhalte, Methoden und Ergebnisse anderer Fächer zurück. Bei dieser elementaren Kooperationsform wird eine Vernetzung von Wissensbeständen für das Verständnis komplexer Problemkreise angestrebt. Weiter gehende Formen erfordern bei ausgewählten Themenfeldern eine wechselseitige Abstimmung des jeweiligen Unterrichts aufeinander oder sogar die gemeinsame Planung einer thematisch bestimmten Unterrichtssequenz durch mehrere beteiligte Fächer.

Bei der Auswahl geeigneter Formen des fachübergreifenden Lernens lassen sich jenseits des „45-Minuten-Taktes“ neue Organisations- und Arbeitsformen vorstellen, die man am ehesten als Studien- und Projektphasen charakterisieren kann.

2. Bereiche, Themen, Gegenstände

2.1 Bereiche: Herleitung und didaktische Funktion

Die Bereiche des Schulfaches Informatik lassen sich nach den Ausführungen in Kapitel 1 auf drei Ebenen eines „didaktischen Würfels“ projizieren. Diese Darstellung erlaubt es, das Schulfach Informatik aus verschiedenen Perspektiven zu beleuchten. Im konkreten Unterricht sind immer alle drei Ebenen im Blick zu behalten und mit Gewinn miteinander zu verknüpfen.

Anwendungssituationen sind häufig der Ausgangspunkt für informatische Entwicklungsprozesse. Sie liefern Anstöße, um sich intensiver mit informatischen Fachinhalten zu beschäftigen und zu theoretischen Einsichten zu gelangen. Die gewonnene Theorie in Form von neuen Fachinhalten erlaubt ihrerseits die Bearbeitung neuer Anwendungssituationen. Die konkrete unterrichtliche Umsetzung ist ein komplexer Vorgang, der insbesondere bei der didaktischen Reduktion detailreicher Anwendungssituationen gleichzeitig die fachorientierten Methoden und die Fachinhalte in den Vordergrund rückt. Im Folgenden werden die Sichtweisen auf die zugehörigen Ebenen des „didaktischen Würfels“ dargestellt.

2.2 Zuordnung der Themen und Gegenstände zu den Bereichen

2.2.1 Fachliche Inhalte

Wenn man den Arbeitsbegriff *Modell* in einer schulnahen - wenn auch wissenschaftlich eher schlichten - Weise benutzen darf als Abbildung eines Ausschnitts der Wirklichkeit bzw. realer Systeme in den Raum zugänglicher Lösungen, als Idealisierung, die nur die wesentlichsten Eigenschaften eines Originals berücksichtigt und als Mittel zum Problemlösen unter den notwendigerweise vereinfachenden Bedingungen von Unterricht, dann lassen sich die Fachinhalte des Informatikunterrichts nach den Ausführungen des vorangehenden Kapitels als Fachprinzipien gebündelt in den zwei Feldern *Modellieren und Konstruieren* sowie *Analysieren und Bewerten* beschreiben:

Lernen im Kontext

Methoden und Formen selbstständigen Arbeitens

Fachliche Inhalte

Modellieren und Konstruieren	**Analysieren und Bewerten**
Ein Informatikmodell gewinnen: Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren	Typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen
Daten und Algorithmen abstrahieren	Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen
Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln	Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennenlernen und einordnen

Die Felder abstrahieren von sehr konkreten und häufig werkzeugabhängigen Fachinhalten zugunsten genereller fachmethodischer Leitlinien, um herauszustellen, dass ein kanonischer Weg durch die informatischen Arbeitsfelder nicht zwingend beschreibbar ist. Vielmehr soll deutlich gemacht werden, dass Einstiege nach diesen Leitlinien an vielen Stellen eröffnet sind, dass unterschiedliche Wege durch die Felder gangbar sind und dass ein Wechsel zwischen ihnen möglich und häufig auch erforderlich erscheint. Einzelne Wege können auf einem höheren Niveau mit zunehmender Selbstständigkeit, mit wachsendem Abstraktionslevel und mit einer strengeren formalen Präzision nach Bedarf iteriert werden. Eine lineare Abfolge von Fachinhalten im Sinne unterrichtlicher Sequenzierung wird hier bewusst nicht beschrieben: entsprechende Hinweise finden sich aber im dritten Kapitel.

Konkrete Feinthemen auf fachpraktischem und fachwissenschaftlichem Niveau lassen sich den Feldern in vielfacher Weise zuordnen. Diese Zuordnungen sind geprägt u.a. von den gewählten Wegen und den konkreten Werkzeugen, dem erreichten Präzisionsgrad und einer mehr reflektierend, einer mehr praktischen oder mehr theoretischen Ausrichtung des jeweiligen Zugangs. In einer vertikalen Gliederung wird der Blick von den Bezügen zur konkreten Lebenswelt und zur Praxis über die typischen Werkzeuge und Methoden des Faches zu den zum Einsatz kommenden Soft- und Hardwarehilfsmitteln gelenkt.

Wenn im Folgenden die Unterpunkte weiter aufgegliedert und erläutert werden, muss nochmals darauf hingewiesen werden, dass dadurch keine unterrichtliche Abfolge beschrieben wird.

2.2.1.1 Modellieren und Konstruieren

Ein Informatikmodell gewinnen:
Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren
- Problemstellungen eingrenzen und Probleme strukturieren
- Anforderungen an ein Modell aufstellen
- ein reduziertes Modell für die Problemstellung definieren
- eine erste Lösungsstrategie entwerfen

Aufgabenstellungen für informatische Aktivitäten ergeben sich in aller Regel nicht aus innerfachlichen Problemstellungen, sondern werden von außen, z. B. von anderen Wissenschaften, gesellschaftlichen Gruppen oder Auftraggebern aus der Wirtschaft formuliert.

Auch für den Informatikunterricht geht man daher von motivierenden, allgemein formulierten Problemstellungen aus, die zunächst auf eine schülergerechte, mit Schulcomputersystemen lösbare Problemstellung eingegrenzt werden. Allem voran müssen die Anforderungen an das zu entwickelnde Informatiksystem präzise definiert werden. Im ersten Lösungsentwurf werden das Ein- und Ausgabeverhalten, die funktionellen Anforderungen, die Qualitätsanforderungen, die ergonomische Gestaltung und das Benutzerverhalten gegliedert und beschrieben. Am Ende dieses Prozesses lässt sich ein Modell für die Problemstellung angeben, welches das konzeptionelle Gerüst des Informatiksystems beschreibt. Lösungsideen werden gesammelt und erste Vorstellungen von groben Lösungsstrategien entwickelt. Im Anfangsunterricht werden nur kleine Probleme mit starken Einschränkungen zu bearbeiten sein, die aber mit zunehmender Beherrschung der Werkzeuge anwachsen.

Die herkömmlichen Grenzen zwischen der hier beschriebenen Systemanalyse und Systemmodellierung, dem eigentlichen Systementwurf und der programmtechnischen Umsetzung werden durch moderne Entwicklungen wie z. B. der Objektorientierung durchlässig, da die relevanten Gegenstände (Objekte und ihre Beziehungen) in vielen Phasen gleich sind und der Entwicklungsprozess spiralförmig verläuft. Bei einem geplanten Einsatz derartiger Werkzeuge sind entsprechende Ansätze frühzeitig bei der Analyse und dem Modellbildungsprozess zu berücksichtigen.

Daten und Algorithmen abstrahieren
- allgemeine Strategien und Standardlösungen kennen lernen und anwenden
- Programmierkonzepte allgemeiner und spezieller Art verstehen und benutzen
- Formen des Strukturierens einsetzen
- problembezogene Objekte und ihre Wechselwirkungen spezifizieren
- ein Lösungskonzept als Denkschema entwickeln

Ist ein zu lösendes Problem hinreichend präzise beschrieben, kann mit der Erarbeitung einer konstruktiven Lösung begonnen werden. Abhängig von den im Unterricht eingesetzten Programmiersystemen kann ein imperatives, objektorientiertes, wissensbasiertes, funktionales oder datenbankgestütztes Lösungskonzept als Denkschema entwickelt werden.

Dazu gehört in aller Regel die Zerlegung der Problemstellung in sinnvolle Teilprobleme und die Entwicklung geeigneter Datenstrukturen zur Aufbereitung und effizienten Verknüpfung der vorgefundenen Informationen. Entsprechende Strukturierungskonzepte müssen erarbeitet werden. Die Lösung der Teilprobleme in Form von Modulen läuft auf die Verwendung oder Entwicklung geeigneter abstrakter Datentypen und der zugehörigen Verarbeitungsalgorithmen hinaus. Auch wenn im konkreten Unterricht unterschiedliche Programmiersysteme eingesetzt werden, lassen sich allgemeine algorithmische und datenorientierte Grundstrukturen herausarbeiten und zugehörige Beschreibungsformen zu ihrer Darstellung entwickeln. Spezielle Konzepte, die vom gewählten Programmierparadigma abhängen, ergänzen diese.

Techniken der (formalen) Spezifikation von Modulen, Prinzipien der Datenabstraktion durch Datenobjekte wie z. B. Datei, Liste, Baum u.a. heben von der speziellen Lösung ab und betonen den universellen Werkzeugcharakter dieser Konzepte. Dazu trägt auch das Herausarbeiten von Klassen algorithmischer Grundstrategien und von Standardlösungen für häufig auftretende Problemstellungen bei.

Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

- Lösungskonzepte implementieren und testen
- Lösungen dokumentieren
- Lösungen nach vorgegebenen Kriterien bewerten
- Problemlösungen optimieren und weiterentwickeln

Die Arbeitsweise der Informatik ist von der Entwicklung und dem Gebrauch leistungsstarker Werkzeuge geprägt. Eines dieser Werkzeuge ist eine Programmiersprache und die zugehörige Entwicklungsumgebung. Deren Einsatz beeinflusst den Informatikunterricht methodisch und inhaltlich an vielen Stellen. Manche Zielsetzungen des Informatikunterrichts werden erst durch den Einsatz derartiger Werkzeuge angemessen erreichbar, andere besser durch deutlich ausgedünnte, didaktisch reduzierte Werkzeuge. Noch so attraktive Werkzeuge dürfen nicht dazu führen, dass deren Bedienung zur Richtschnur des Unterrichts wird. Vielmehr ist immer darauf zu achten, in die Bedienung und in die Sprachelemente nur so weit einzuführen, wie es zur Beschreibung der Lösung erforderlich ist.

Systematische Tests sind notwendig, um zu einem korrekten Programm zu kommen. Im System angebotene Testhilfen sollten genutzt, Testdaten ausgewählt und festgelegt werden. Die lauffähige Gesamtlösung und der gesamte Lösungsprozess sind zu dokumentieren, wobei die gewählten Verfahren und die verwendeten Hilfsmittel sowie die erforderlichen Informationen für die Benutzer beschrieben werden. Bei Verfügbarkeit geeigneter Hilfen zur entwicklungsbegleitenden Dokumentation sollten diese genutzt und von Beginn an in die Entwicklungsarbeit einbezogen werden. Die erreichte Gesamtlösung ist mit der ursprünglichen Problemstellung, der Spezifikation und den dort entwickelten Systemkriterien zu vergleichen und zu bewerten. Eventuell ist eine Modifikation (Einschränkung, Erweiterung, Neukonzeption) der Problemlösung notwendig und der gesamte Entwicklungsprozess zu wiederholen.

2.2.1.2 Analysieren und Bewerten

Typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen

- die Entwicklung von Informatiksystemen kennen lernen und verstehen
- den Einsatz von Informations- und Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten
- den Strukturwandel in Industrie und Gesellschaft erkennen und beschreiben
- die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen

Typische Einsatzbereiche von Informations- und Kommunikationssystemen und die Auswirkungen des Computereinsatzes können an exemplarischen Anwendungen analysiert und beurteilt werden. Der Einfluss der Datenverarbeitung auf Entscheidungs- und Arbeitsabläufe soll dabei beispielhaft aufgezeigt werden. Mit in den Blick zu nehmen ist, wie weit einzelne Menschen oder die Gesellschaft durch den Gebrauch oder durch den möglichen Missbrauch der Datenverarbeitung betroffen sind.

Die in der Praxis genutzten Informatiksysteme sind derartig umfangreich, spezialisiert und in rascher (Weiter)Entwicklung begriffen, dass die Kennzeichnung typischer Einsatzbereiche nur im Überblick erfolgen kann. Exemplarisch können einige Anwendungsbereiche tiefergehenden Analysen unterzogen werden. Dabei wird man sich auf schriftliches Material, eventuell ergänzt durch Unterrichtsgänge, beschränken. Teilweise ist der Einsatz von Demonstrationsversionen kommerzieller Software sinnvoll. Ausgewählte Anwendungssituationen erlauben es auch, eigene, reduzierte Modelle zu konstruieren.

Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen

- den Algorithmenbegriff und den Begriff der Berechenbarkeit verstehen
- Grenzen von Verfahren und Methoden abschätzen
- formale Sprachen und Grammatiken untersuchen
- Syntaxregeln und Beschreibungssysteme beurteilen
- Automatenmodelle und akzeptierte Sprachen analysieren und beurteilen
- Effizienzuntersuchungen durchführen

In das Zentrum der theoretischen Grundlagen der Informatik gehört die Präzisierung des Algorithmusbegriffs und das Erkennen der Grenzen der Algorithmisierbarkeit. Die Fragen nach den Grenzen der Berechenbarkeit führen schließlich darauf, was Computer prinzipiell können und was sie prinzipiell nicht können.

Programmiersprachen sind Beispiele formaler Sprachen, diese werden durch Grammatiken oder entsprechende symbolverarbeitende Automaten spezifiziert. Syntaxregeln, Beschreibungssysteme und Signaturen spielen bei allen Formalisierungsprozessen eine wesentliche Rolle, Automatenmodelle können den Modellierungsprozess wesentlich unterstützen. Einerseits kann dadurch das grundlegende, leistungsfähige Konzept symbolischer Informationsverarbeitung herausgearbeitet werden, andererseits lässt sich die Beschränktheit formaler Kommunikation zeigen und zum menschlichen Sprechen, Denken und Handeln in Beziehung setzen.

Die Effizienzuntersuchung beschäftigt sich mit der Frage, mit welchem Aufwand an Rechenzeit und/oder Speicherplatz bestimmte algorithmische Aufgabenklassen gelöst werden können und ob deren Lösbarkeit an praktische Grenzen stößt. Sie bewahrt davor, mit unrealistischen Ansätzen und überzogenen Erwartungen an Probleme heranzugehen und dadurch Ressourcen zu verschwenden.

Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen

- die Struktur und Funktionsweise eines von-Neumann-Rechners analysieren
- alternative Rechnerkonzepte und Maschinenmodelle beschreiben
- die Informationsdarstellung auf der Maschinenebene analysieren
- Funktionen und Komponenten der Systemsoftware kennen lernen u. beschreiben
- Kommunikations- und Vernetzungsstrukturen einordnen
- Anwendungssoftware klassifizieren
- Benutzerführung, Funktionsumfang und Schnittstellen untersuchen und bewerten

Die im Unterricht eingesetzten Hard- und Softwaresysteme werden als symbolverarbeitende Maschinen analysiert. Die sich dahinter verbergenden Ideen sind so transparent zu machen, dass die prinzipielle Wirkungsweise technischer oder softwaremäßiger Produkte klar wird.

Auf der untersten Ebene sind dies die Prinzipien der Digitalisierung und der binären Codierung von Daten und Befehlen. Die dazugehörigen Bitmuster sind sprachliche Konstrukte einheitlicher Struktur, lassen sich aber in verschiedenen Bedeutungen von einer Rechenmaschine interpretieren. Im historischen Entwicklungsprozess wird die von Neumannsche Idee der Informationsverarbeitung durch programmgesteuerte Automaten und der zugehörigen Rechnerarchitekturen thematisiert. Alternative Konzepte bis hin zu lokalen und globalen Vernetzungsstrukturen können dieses Thema sinnvoll ergänzen.

Typische Sprachelemente einer höheren, problemnahen Programmiersprache werden exemplarisch auf ein elementares maschinennahes Niveau transformiert. Damit wird der nach wie vor aktuelle Prozess, zu immer leistungsfähigeren, d.h. problemnäheren Programmiersprachen mit noch mächtigeren Sprachkonstrukten zu kommen, beispielhaft umgekehrt und aufgezeigt, dass die volle Rechenleistung einer Maschine bereits auf sehr niedrigem Sprachniveau vorliegt.

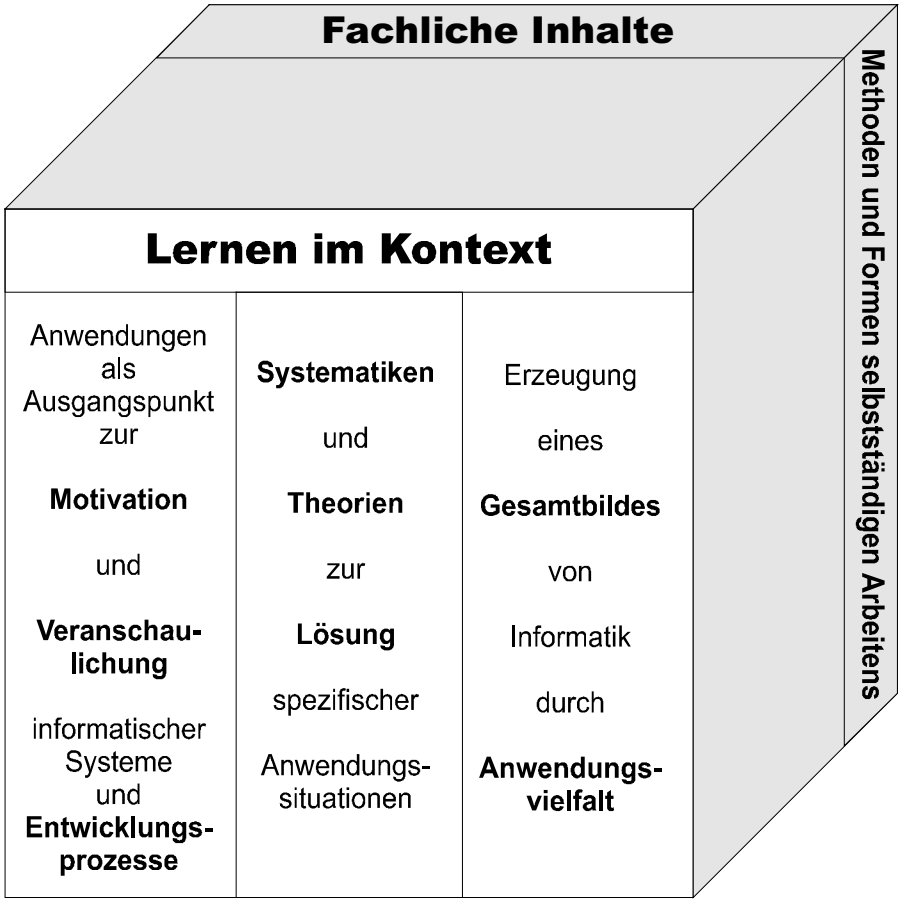
Funktionen und Komponenten der Systemsoftware werden in einer stark vereinfachten Darstellung herausgearbeitet. Daran kann verdeutlicht werden, dass Rechner ihre Leistungsfähigkeit erst durch den Verbund untereinander in Wechselwirkung stehender Teilsysteme entfalten können und sich zu immer größeren, leistungsfähigeren Systemen zusammenschließen.

Zusätzlich wird einsichtig, dass der Rechner mit Hilfe „elementarer“ Programme z. B. die Transformation von der höheren auf die maschinennahe Sprachebene selbst übernehmen kann. Aspekte der theoretischen Informatik können hier helfen, grundlegende Ideen bewusst zu machen und von den unwesentlichen technischen Einzelheiten zu trennen. Modellsysteme mit didaktisch reduziertem Funktions- und Befehlsumfang lassen sich sinnvoll einsetzen, um Grundprinzipien zu veranschaulichen. Ein Übergang zur Entwicklung eigener, exemplarischer Automatenmodelle ist möglich.

Aus den so gewonnenen Kenntnissen sind die Funktionsprinzipien der im Unterricht verwendeten Anwendungssoftware ableitbar; die Benutzerführung, der Funktionsumfang und die Schnittstellen lassen sich bewerten und mit anderen Softwarelösungen vergleichen.

2.2.2 Lernen im Kontext

Innerhalb der verschiedenen Sichtweisen der Schulinformatik hat die Ebene der Anwendungen in dreifacher Hinsicht eine große Bedeutung:



Die Hereinnahme konkreter informatischer Anwendungen in das Unterrichtsgeschehen ermöglicht einerseits, die Schülerinnen und Schüler auch längerfristig für eine Lernsequenz zu motivieren, und andererseits, Zusammenhänge, Gesetzmäßigkeiten und Theorien an konkreten Objekten, Systemen bzw. Prozessen zu veranschaulichen. Gerade in der Einbeziehung der Anwendungsebene unterscheidet sich die Schul- von der Hochschulinformatik, die die primäre Aufgabe in der fachwissenschaftlich-theoretischen Aufarbeitung und Darstellung ihrer Themen sieht und weniger um das Begreifen und den Lernprozess bei ihren Adressaten bemüht ist als die Schule.

Fachinhalte unter den Aspekten *Modellieren und Konstruieren* sowie *Analysieren und Bewerten* zu bearbeiten heißt unverzichtbar, Lernende mit Anwendungssituationen zu konfrontieren und entweder schon bekannte Methoden und Kenntnisse zur modellhaften Bewältigung

des Problems anzuwenden oder durch Abstraktion, Verallgemeinerung und Systematisierung zu diesen Methoden und Fachkenntnissen zu gelangen. Schülerinnen und Schüler können nur in realen bzw. modellhaft nachempfundenen Anwendungssituationen eine Informatik-Kompetenz erwerben, d.h. informatische Probleme durchschauen, systematisch lösen und in ihren Anwendungs- und Verwertungszusammenhängen richtig beurteilen.

Schließlich lässt sich eine facettenreiche Sicht der Informatik, ihrer Bedeutung und Auswirkungen nur erreichen, wenn vielfältige Bezüge zu unterschiedlichen Anwendungsfällen hergestellt werden und sich diese bei den Adressaten allmählich zu einem Bild der Informatik zusammenfügen.

Ein handlungs- und produktorientierter Unterricht wird deshalb solche informatischen Anwendungen aus unserer Lebenswelt aufgreifen, die exemplarischen Charakter haben und tiefere Einsichten in die gedanklichen Prozesse der Informatik und ihrer Auswirkungen ermöglichen.

Das soll und kann jedoch nicht heißen, dass Anwendungen - z. B. der Umgang mit einer Textverarbeitung oder einem Datenbanksystem - um ihrer selbst willen betrieben werden und sich der Unterricht weitgehend darin erschöpft, mit Standardsoftware zu arbeiten und ihre Nutzung zu perfektionieren. Vielmehr sind die informatischen Anwendungen der geeignete Nährboden, um die Fachinhalte daraus zu abstrahieren, zu verallgemeinern und die fachlichen Methoden des analytischen und konstruktiven Herangehens an Problemlösungen bzw. zu lösende Probleme einzuführen und immer wieder zu üben. Ein richtig verstandener Informatikunterricht in der gymnasialen Oberstufe wird die Prinzipien und theoretischen Grundlagen herausarbeiten und diese stärker betonen als vordergründige Anwendungen, die heute wichtig erscheinen, morgen jedoch vielleicht schon überholt sind.

Als typische Anwendungsbereiche, die exemplarischen Charakter haben und sehr unterschiedliche Facetten der Informatik beleuchten, kommen für den Unterricht u.a. in Betracht:

Informationssysteme

Hier lassen sich Abfrage- und Auskunftssysteme wie z. B. die Fahrplanauskunft behandeln und der Aufbau von Datenbanken konzeptionell beleuchten (relationale, objektorientierte Datenbankmodelle).

Messen, Steuern, Regeln

Im Mittelpunkt stehen Aspekte der digitalen Signalverarbeitung, und zwar weniger in technischer als vielmehr in algorithmischer Form. Wie wird z. B. ein Plotter, der mit Schrittschaltmotoren ausgerüstet ist, digital vom Computer angesteuert? Nach welchem Algorithmus funktioniert z. B. ein Fahrkartenautomat, wie kann sein Verhalten formal (automatentheoretisch) beschrieben und dann durch Programme simuliert werden?

Computersimulation

Simulationen mit Hilfe des Computers werden häufig zu Planungs- und Prognosezwecken benutzt, z. B. zur Vorhersage des Bevölkerungswachstums, der langfristigen Entwicklung des Erdklimas, oder auch zur Optimierung von Systemen, die vor ihrer Realisierung mit verschiedenen Parametern im Modell durchgespielt werden. An einfachen Beispielen - *Wie viel Zapfsäulen muss eine Tankstelle haben, damit kein Kunde mehr als 2 Minuten warten muss?* - lässt sich das Prinzip der ereignisgesteuerten Simulation herausarbeiten und in unterschiedlichen Schwierigkeitsgraden programmtechnisch realisieren. Die Modellbildung wird hier konkret als dynamischer, vielfach rückgekoppelter Prozess erlebt.

Bildverarbeitung, Mustererkennung

Die Bearbeitung grafischer Daten durch Computer bzw. die grafische Interpretation von Zeichen und Zeichenmustern ist zu einem der wichtigsten Anwendungsgebiete der Informatik geworden. Ausgehend von einem realen CAD-System und seinen mächtigen Grafikbefehlen schärft eine einfache Modellierung den Blick für die Probleme der Datenrepräsentation, der zwei- und mehrdimensionalen Darstellung auf dem Monitor und für die umfangreichen mathematischen Verarbeitungsalgorithmen.

Das Problem der Mustererkennung steckt in vielen Anwendungen und wird stets prinzipiell ähnlich gelöst, sei es beim optischen Scannen von Identnummern (Ausweise, Warencodes), beim Scannen von Textseiten als Bild, um sie anschließend in Textform überführen und als Text weiterverarbeiten zu können, oder bei der automatischen Steuerung von Flugkörpern, die als digitale Muster gespeicherte Landschaften mit Bildern vergleichen, die Fernsehkameras während des Fluges liefern.

Telekommunikation

Die Vernetzung von Computern und lokalen Datennetzen ermöglicht einen Datenaustausch in bisher nicht gekanntem, stark steigendem Ausmaß. Ausgehend vom Datenverkehr in einem lokalen Netz und der Anbindung ans Internet können die Prinzipien des Datenaustausches und der technikgestützten Kommunikation, ihre verschiedenen Modelle und zusätzlichen Randbedingungen vermittelt werden. Bei diesem Thema sollte auf die möglichen gesellschaftlichen Veränderungen aufmerksam gemacht werden, die von dieser Technologie ausgehen und die damit zu tun haben, dass Information für alle - lokal und global - leicht verfügbar wird. Erste Erfahrungen lassen sich auf diesem Gebiet durch eigenes Tun sammeln, wenn z. B. einfache Übertragungsprotokolle für die serielle oder parallele Schnittstelle von PCs entwickelt und ausprobiert werden.

Sprachverarbeitung

Es ist nicht erstaunlich, dass Sprachwissenschaftlerinnen und Sprachwissenschaftler sich seit langem informatisch betätigen und Informatikerinnen und Informatiker Erkenntnisse der Sprachwissenschaften verarbeiten. Der Abstand zwischen der Verarbeitung einer natürlichen

Sprache (Verstehen, Übersetzen) und dem Umgang mit einer künstlichen formalen Sprache, wie es die Programmiersprachen sind, wird immer geringer. Ein Compiler folgt bei der Übersetzung einer Programmsequenz in die digitale Maschinensprache ähnlichen algorithmischen Regeln wie ein elektronischer Übersetzer im Taschenrechnerformat, der einen eingetippten deutschen Satz in Englisch oder Französisch anzeigt. Die Regeln formalsprachlicher Verarbeitung lassen sich am Beispiel des Compilers gut herausarbeiten und verallgemeinern, die Grenzen zu lebendigen Sprachen werden in dem Spannungsfeld zwischen Syntax und Semantik deutlich.

Strategiespiele

Spiele, die strategische Überlegungen erfordern, also Brettspiele wie Mühle, Dame, Schach etc. eignen sich gut, in modellierten spielerischen Situationen den Verlauf von Entscheidungsprozessen nach einem Regelwerk zu steuern. Wenn auch zunächst das konkrete Spiel im Mittelpunkt der Betrachtungen steht, so lässt sich doch zeigen, dass die im Spiel verwendeten Entscheidungsmechanismen sich verallgemeinern und auf „ernsthafte“ Problemstellungen übertragen lassen.

Datenschutz und Datensicherheit

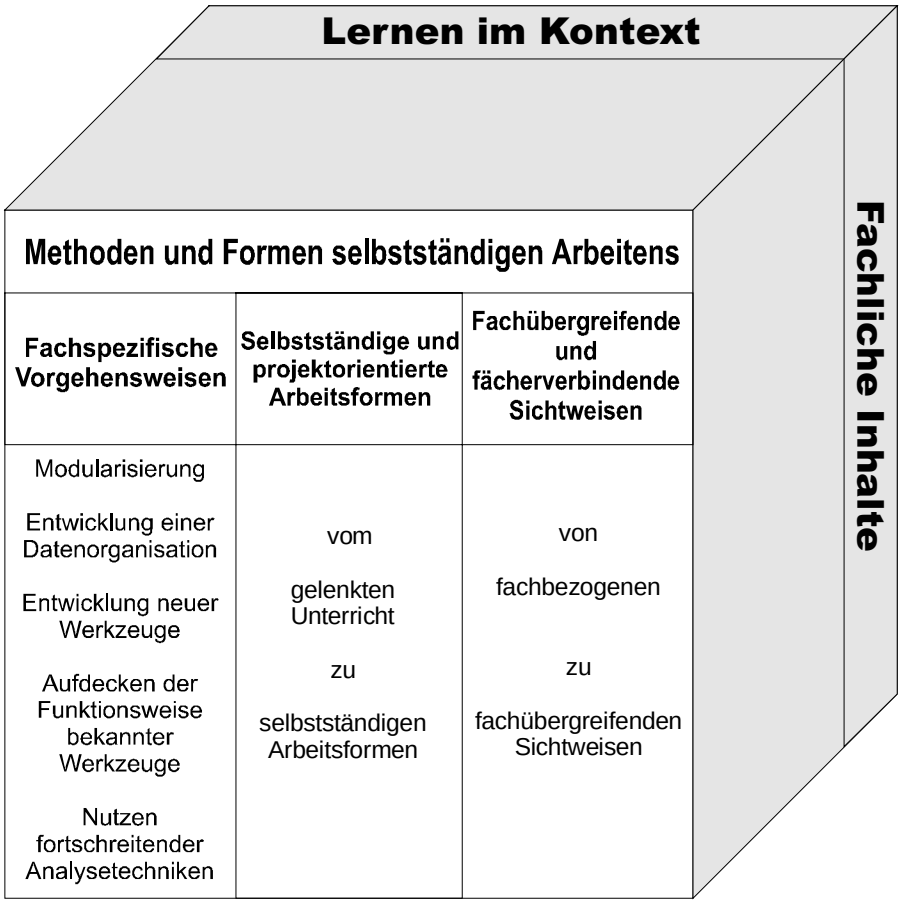
Zu diesem Thema gehören Übertragungsverfahren und kryptologische Verschlüsselungsalgorithmen, die zunächst stark mathematikbetont zu sein scheinen, aber innerhalb informatischer Anwendungen aus diesem Themenbereich wie z. B. Telebanking und Teleshopping nur einen Aspekt unter mehreren darstellen. Neben den kryptologischen und datentechnisch-organisatorischen Grundlagen wird zwangsläufig auf die politisch-gesellschaftlichen Dimensionen eingegangen, z. B. auf den Konflikt zwischen dem Recht des Individuums auf unversehrte Übermittlung seiner Daten und dem Staatsinteresse, die Verschlüsselung von Daten im Übertragungsfall zu Kontrollzwecken einzuschränken.

Die Aufgabenstellungen für den Unterricht werden realen Anwendungsbereichen, möglichst der Lebens- und Erfahrungswelt der Jugendlichen entnommen. Das führt notwendigerweise dazu, den konkreten Anwendungszusammenhang handhabbar zu reduzieren, im Modell zu bearbeiten und das Ergebnis wieder an der Realität zu messen.

Im Informatikunterricht verbinden sich also in besonderer Weise die Ebene der Anwendung, die Arbeitsweise der Modellbildung und fachübergreifende Aspekte, die zusammen im projektartigen Arbeiten ihre adäquate Methode finden.

2.2.3 Methoden und Formen selbständigen Arbeitens

Methoden werden im ursprünglichen Wortsinn als Wege zu einem Ziel verstanden. Im engeren Sinne einer Didaktik des Schulfaches Informatik, wie sie sich aus der Sicht des vorgestellten Würfels darstellt, werden hier im Wesentlichen drei Aspekte von Methoden in den Blick gerückt. Der Erste bezieht sich auf fachspezifische Vorgehensweisen, der Zweite auf die organisatorische Rolle der Unterrichtenden als Vermittler und beschreibt den Weg vom gelenkten Unterricht zu selbständigen und projektorientierten Arbeitsformen, und der Dritte die Entwicklung von fachbezogenen hin zu fachübergreifenden Sichtweisen.



2.2.3.1 Fachspezifische Vorgehensweisen

Ohne grundlegende informatikspezifische Methoden lassen sich Modelle zu gegebenen Aufgabenstellungen kaum entwickeln. Da diese in der Regel zu komplex sind, um in einem einzigen Arbeitsgang erfasst oder gar gelöst zu werden, ist als erster Schritt das Problem in einfachere Teilprobleme zu zerlegen und bis auf die zu beschreibenden Schnittstellen in voneinander unabhängige Teilaufgaben aufzuteilen (Prinzip der Modularisierung).

Der nächste Schritt ist dann die Wahl einer geeigneten Datenorganisation in der verwendeten Programmierumgebung. Diese gibt ihrerseits die Werkzeuge vor, mit denen eine algorithmische Lösung der einzelnen Module erarbeitet und realisiert werden kann. Dabei wird man sich zunächst auf die in der Programmiersprache vorgegebenen Möglichkeiten beschränken und ihre Funktionsweisen aufdecken, um sie im Rahmen der Aufgabenlösung sinnvoll einsetzen zu können.

Mit fortgeschrittener Erfahrung in der Lerngruppe wird man dazu übergehen, aus den gegebenen Standards der Sprache kreativ neue und leistungsfähigere Datentypen und Werkzeuge zu entwickeln - im klassischen Sinne wären das allgemeine (abstrakte) Datentypen. Methodisch wird dies auf natürliche Weise durch komplexere Aufgabenstellungen erreicht, die mit elementaren Objekten und Werkzeugen nicht problemadäquat zu bewältigen wären. Die weiteren methodischen Schritte sind die Analyse und kritische Bewertung der Problemlösung und der Funktionalität des Programms mit ggf. einer entsprechenden Modifizierung.

2.2.3.2 Selbstständige und projektorientierte Arbeitsformen

Die unterrichtlichen Arbeitsformen unterscheiden sich generell nicht von denen, die auch in anderen Fächern Verwendung finden und sich aufteilen lassen in die Sozialformen des gemeinsamen Lehrens und Lernens im gesamten Kurs, des arbeitsteiligen Vorgehens in einzelnen Gruppen und der Einzelarbeit. Der Lehrervortrag, das gelenkte, das fragend-entwickelnde und das freie Unterrichtsgespräch behalten auch in der Informatik ihren Platz. Einzelarbeit, wie sie etwa in der Präsentation von Hausaufgaben, Referaten oder Facharbeiten zum Ausdruck kommt, offenbart die Fähigkeit des Einzelnen *zu selbständigem Arbeiten und Problemlösen* und liefert somit wesentliche Beiträge zur individuellen Bewertung.

Der in Kapitel 1 skizzierte Problemlösungs- und Entwicklungsprozess wird in der Informatik auf besondere Weise durch zunehmend komfortabler und benutzerfreundlicher werdende Werkzeuge der Hard- und Software erleichtert: Gute Anwendungsprogramme oder Programmierumgebungen mit entsprechenden Online-Hilfen und Fehlermeldungen nehmen den Unterrichtenden zumindest im formal-syntaktischen Bereich einen Teil ihrer Korrekturarbeit ab. Die Schülerinnen und Schüler können in weiten Teilen selbständig ihre Erfahrungen bei der Umsetzung ihrer Problemlösungen machen und eine direkte Kontrolle und Hilfestellung durch ihren maschinellen „Partner“ erfahren.

Das selbständige Arbeiten bezieht sich allerdings nicht nur auf individuell erbrachte Leistungen sondern ebenso auf die gemeinschaftlichen Beiträge von Arbeits- und Entwicklungsgruppen. Das Fach Informatik ist besonders im *projektorientierten Gruppenunterricht* auf die vielfältigen Möglichkeiten partnerschaftlicher Arbeitsformen angewiesen. Dies korrespondiert am ehesten mit der Komplexität der Aufgabenstellungen, die aus den Anwendungsbereichen an die Informatik herangetragen werden, und die häufig den motivierenden Ausgangspunkt für eine unterrichtliche Aufarbeitung bilden.

Dabei steht nicht so sehr die Orientierung an markt- und konkurrenzfähigen Produkten im Vordergrund als vielmehr die Erfahrung, dass in unserer zunehmend komplexen Lebenswirklichkeit der Einzelne nicht immer alles allein bewerkstelligen kann. Das kooperative Arbeiten innerhalb der Gruppen, das Zusammenwirken der einzelnen Teams, die Verlässlichkeit der Absprachen und das Bewusstsein um die Verantwortung für das Gelingen des Ganzen erhalten dabei ein starkes Gewicht. Dabei darf der Einzelne das Gesamtziel nicht aus dem Auge verlieren, er muss wissen, was in den anderen Gruppen passiert und wie die eigene Leistung sich mit denen der anderen zu einer Komplettlösung zusammenfügen soll.

Das schulische Unterrichtsprojekt dient - anders als in der Arbeitswelt - nicht in erster Linie der Nutzenmaximierung und Effizienzsteigerung, sondern macht den Sinn gemeinsamen

Schaffens und die Freude an sozialintegrativen Arbeitsformen erlebbar. Hierbei kommt den Unterrichtenden bei behutsamer Moderation eine besondere Verantwortung zu, insbesondere dann, wenn die Ziele der Gruppe sich im Verlaufe der Arbeit als unerreichbar hoch gesteckt erweisen. Konkrete Anregungen und Beispiele zu Projektunterricht finden sich in Kapitel 3.

2.2.3.3 Fachübergreifende und fächerverbindende Sichtweisen

Der Anfangsunterricht in Informatik ist durch *fachspezifische* Gegenstände geprägt wie das Erlernen von Elementen einer geeigneten Sprache und die Einarbeitung in eine Programmierumgebung, in der sich Lösungen angemessen formulieren lassen. Dabei stellt sich bald heraus, dass die Aufgaben, für die man algorithmische Lösungen sucht, aus ganz unterschiedlichen Fachbereichen stammen. Insofern fordert die Informatik eine *fächerverbindende* und *fachübergreifende* Betrachtungsweise geradezu heraus. Grundsätzliche Gedanken hierzu sind bereits in Kapitel 1.2 dargestellt.

Durch eine Vielzahl von Algorithmen hat die Informatik inzwischen erhebliche Dienstleistungen für andere Fächer erbracht. Hierzu gehören mathematische Berechnungen, Simulationen von Zufallsprozessen und dynamischen Systemen, Verfahren der Linguistik, der künstlichen Intelligenz und der Umgang mit formalen Sprachen. Der Bezug zu anderen Fachbereichen ist also unmittelbar vorhanden und muss nicht erst künstlich hergestellt werden.

Angesichts der Verzahnung mit anderen Bereichen erscheint es im Gegenteil eher notwendig, im Laufe des Unterrichts das spezifisch Informatische als eigenständigen fachlichen Kern herauszuarbeiten. Auf die Algorithmik bezogen bedeutet das etwa, den Beitrag der Informatik zur Entwicklung von Strategien des Denkens hervorzuheben und diese zu bewerten.

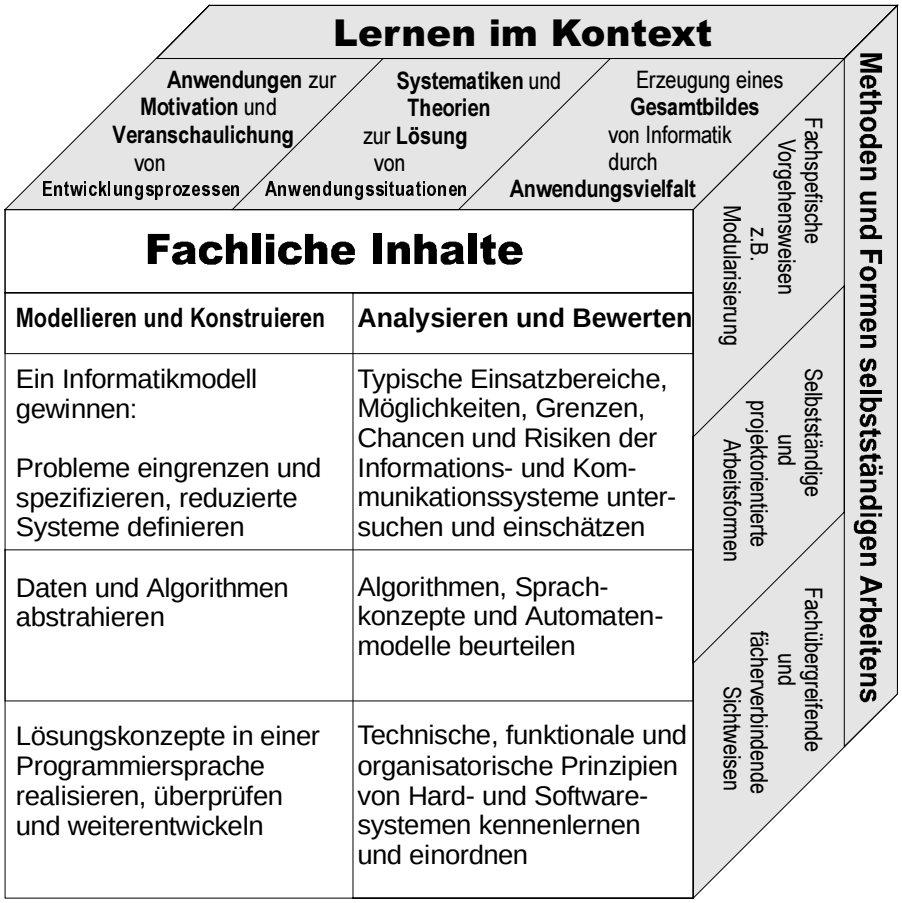
Zusammenfassend lässt sich feststellen: Der fachübergreifende Aspekt der Informatik liegt in ihren Anwendungsfeldern. Der fächerverbindende Aspekt ergibt sich durch die allgemeinen Methoden, die bei Problemlösungen in verschiedenen Fachbereichen benutzt werden.

Konkrete Anregungen und Beispiele - auch zu möglichen Organisationsformen fachübergreifender Projekte - finden sich ebenfalls in Kapitel 3.

2.3 Obligatorik und Freiraum

Die in Kapitel 2.2 ausführlich dargestellten und auf den Flächen des „didaktischen Würfels“ komprimiert abgebildeten Bereiche des Faches mit ihren impliziten und expliziten wechselseitigen Bezügen bilden die verpflichtende Struktur für die Ableitung von Lernsequenzen. Dadurch werden ein verbindlicher Rahmen für die Behandlung von Fachinhalten beschrieben, Anwendungsfelder für das Lernen in größeren Zusammenhängen angegeben und unterrichtliche Methoden für das Fach Informatik zusammengestellt. Die Verbindlichkeit bezieht sich nicht isoliert auf jedes einzelne Stichwort, sondern in erster Linie auf die Berücksichtigung der Struktur als Ganzes. Für die Umsetzung in eine zeitlich linear verlaufende Unterrichtssequenz bleibt damit Freiraum für didaktische Entscheidungen der Lehrerinnen und Lehrer, ohne inhaltlicher Beliebigkeit Raum zu geben.

In der würfelförmigen Übersicht sind die Stichworte noch einmal komprimiert aufgeführt:



Die Ausrichtung auf das Qualifikationsziel Abitur impliziert, dass Flexibilität nicht mit inhaltlicher Beliebigkeit gleichgesetzt werden darf und dass Pointierung und Akzentuierung einzelner Teilbereiche der Informatik dem Erreichen eines qualitativ anspruchsvollen und quantitativ abgerundeten Abschlussprofils verpflichtet bleiben.

2.3.1 Stufung der Obligatorik

Die Obligatorik ist im vorliegenden Lehrplan gestuft dargestellt um sicherzustellen, dass die generellen Vereinbarungen möglichst unabhängig bleiben vom raschen Generationswechsel im Bereich der Programmierwerkzeuge.

(1) Uneingeschränkt obligatorisch sind
- die fachlichen Inhalte mit den in 2.2.1 dargestellten 6 Textkästen zum Modellieren und Konstruieren sowie zum Analysieren und Bewerten,
- das Lernen im Kontext mit den in 2.2.2 dargestellten acht verschiedenen Facetten der Anwendung,
- die Methoden und Formen selbständigen Arbeitens mit den in 2.2.3 dargestellten drei Vorgehensarten, Arbeitsformen und Sichtweisen,
- die in 2.3.2 gemachten Vorgaben zur Konstruktion möglicher Unterrichtssequenzen.

(2) In Abhängigkeit von der Wahl eines der verbreiteten Sprachparadigmen sind in 2.3.3 die obligatorischen Anteile für unterschiedliche Sprachkonzepte entwickelt.

(3) Während unter (1) ein zwingender Kanon beschrieben ist, bilden die in Kapitel 3.4 dargestellten Lernsequenzen Beispiele für eine passende Umsetzung, die als solche natürlich nicht in allen Einzelheiten verbindlich sein können.

2.3.1.1 Jahrgangsstufe 11

Die Grundkurse führen in der Jahrgangsstufe 11 in die Fächer der gymnasialen Oberstufe ein. Thematisch und methodisch werden den Schülerinnen und Schülern die notwendigen Voraussetzungen vermittelt, um den Anforderungen der anschließenden Qualifikationsphase gerecht werden zu können. Von Vorteil erscheint eine curricular spiralförmige Anlage des Unterrichts, bei der wesentliche Bereiche des Faches in ihren Grundzügen in der Jahrgangsstufe 11 unterrichtlich erschlossen werden, um sie in der Qualifikationsphase auf höherem Niveau und in komplexeren Zusammenhängen vertieft aufzuarbeiten bzw. fortzuführen.

Im Mittelpunkt steht das Kennenlernen von *Fachinhalten* und *informatikspezifischen Methoden* an *Beispielen*, die an Erfahrungen und Vorkenntnissen der Schülerinnen und Schüler anknüpfen. Bei den Fachinhalten liegt das besondere Augenmerk zunächst auf dem *Modellieren und Konstruieren*, wobei mehrere Informatikmodelle aus unterschiedlichen Bereichen unter Nutzung einer Programmiersprache /-umgebung von der Problemeingrenzung über die Daten- und Algorithmenabstraktion bis zur Realisierung auf dem Computer entwickelt werden. Anteile aus dem Feld *Analysieren und Bewerten* werden in der Jahrgangsstufe 11 nur eingeschränkt vermittelt.

Die an verschiedene Sprachparadigmen gekoppelten „Einlösungen“ konkretisieren dieses Vorgehen auf fachsprachlichem Niveau.

Bei den Unterrichtsmethoden sollen vermittelnde und informierende Phasen ergänzt werden durch Phasen mit starker Eigentätigkeit: Computergerechte Lösungsvorschläge werden in Gruppen erarbeitet und die zugehörigen Programmentwürfe werden von den Schülerinnen und Schülern selbst implementiert.

Die Anwendungsbeispiele sollen Realitätsbezug haben und an Gegenstände der öffentlichen Aufmerksamkeit anknüpfen.

2.3.1.2 Jahrgangsstufen 12 und 13

In der Qualifikationsphase werden die erworbenen Kenntnisse bezüglich fachlicher Inhalte und Methoden erweitert und vertieft. Die eigenständige Realisierung von komplexeren (Teil)Lösungen ist weiterhin das Ziel im handlungsorientierten Unterricht. Das Feld Modellieren und Konstruieren wird dabei mehrfach durchlaufen, wobei zu den Standardmitteln der benutzten Programmierumgebung selbst entwickelte Konstrukte im Sinne einer Spracherweiterung in den Vordergrund rücken und durch andersartige, weiterführende Ansätze mindestens eines zweiten Paradigmas ergänzt und oder kontrastiert werden.

Eingebettet in den Anwendungsbezug liefert die kritische Einschätzung von Modell-Reduktionen und Lösungsansätzen produktimmanente Beiträge zum Themenfeld Analysieren und Bewerten. Schließlich wird die Entwicklung der Informatik und ihrer Anwendungen im gesellschaftlichen Raum mit wachsenden Kenntnissen fundierter beurteilbar und die eigene Position in diesem Umfeld genauer bestimmt.

2.3.1.3 Zusammenarbeit mit einer Naturwissenschaft

Sollte das Fach im Rahmen einer insgesamt vierstündig unterrichteten Kombination mit einem anderen, nämlich naturwissenschaftlichen Fach angeboten werden, so bleibt die Grundstruktur der Obligatorik erhalten. Der Verlust an fachspezifischer Unterrichtszeit erfordert aber Beschränkungen im informatikspezifischen Themenangebot: zurückzunehmen wären zum Beispiel die Anteile der sog. theoretischen Informatik und der technischen Aspekte der Informationsbearbeitung. Ein Verzicht auf die inhaltliche Verflechtung einzelner Unterrichtsfelder vor einem gemeinsamen theoretischen Hintergrund und auf die Aspekte systematischer Analyse und Bewertung von Informatiksystemen schafft weitere zeitliche Entspannung, bedeutet aber einen weit reichenderen Verlust an fachlicher Identität und Qualität.

2.3.2 Zur Konstruktion möglicher Unterrichtssequenzen

In der ersten „Spiralwindung“ der unterrichtlichen Sequenzierung soll am Ende der Jahrgangsstufe 11 insofern ein thematisch abgerundeter Kenntnisstand erreicht sein, als insbesondere alle Facetten zum Modellieren und Konstruieren vermittelt wurden, die im „didaktischen Würfel“ aufgenommen sind.

Im Laufe der Qualifikationsphase sind dann sämtliche Bereiche des Faches abzudecken, wobei nicht jeder Unterrichtsabschnitt alle Felder des „didaktischen Würfels“ gleichermaßen aufnehmen kann, sondern sinnvolle Schwerpunkte erkennen lässt.

Der gesamte Unterricht in den Jahrgangsstufen 11 bis 13 zeichnet sich nicht durch das ständige Einbringen völlig neuer Prinzipien aus, sondern durch vertieftes theoretisches Durchdringen, durch komplexer werdende Aufgabenstellungen, durch Zuwachs an Selbstständigkeit bei der Bearbeitung und durch fachlich fundiertere Beurteilungen und Stellungnahmen. Insofern hat eine konkrete Folge von Unterrichtsthemen die zunehmende Selbstständigkeit im Umgang mit informatischen Elementen und die wachsende Komplexität von Aufgaben, Lösungen und Bewertungen in den Mittelpunkt zu stellen.

Obwohl jede Sequenzierung der zu lernenden Inhalte und Methoden stark vom Wesen der zugrunde gelegten Programmiersprache beeinflusst wird, sollten folgende Aspekte zu den *fachspezifischen Vorgehensweisen* immer wieder richtungweisend bleiben.

2.3.2.1 Modularisieren und Entwickeln einer geeigneten Organisation

Im Zusammenhang mit Lösungen durch Programme lassen sich hier komplexe Aufgabenstellungen in einfachere, unabhängig zu bearbeitende Teilaufgaben zerlegen. Es werden relevante

Teilgedanken isoliert, spezielle Operationen programmiert, ein Entwurfsdesign zusammengestellt usw.

Die Abbildung der Wirklichkeit im Lösungsansatz führt zum Erkennen standardisierter Datenstrukturen und zum Entwickeln eigener Datenobjekte bzw. Werkzeuge über die Grundstruktur der verwendeten Sprache hinaus. Vorstellungen zum Ablauf eines Programms oder der Zusammenarbeit von Programm-Modulen und der Steuerung im Programm sollen präzisiert, vertieft und kreativ verwendet werden.

2.3.2.2 Aufdecken der Funktion bekannter und Entwickeln neuer Werkzeuge

Zu Beginn werden Sprachkonstrukte vielleicht einfach und einheitlich als „Befehle“ verstanden; Schülerinnen und Schüler akzeptieren den Computer und seine Funktionsweise als „black box“ und nehmen die gelernten Elemente und die Grammatik einer Hochsprache als eng begrenzt und gegeben hin. Mit fortschreitendem Fachunterricht jedoch entwickelt sich das Verständnis für die benutzten Werkzeuge weiter. Ihre Konstruktionsprinzipien, Grundlagen und Funktionsweisen werden Schritt für Schritt aufgedeckt, das Gesamtverständnis erweitert sich.

Im Informatikunterricht werden Lösungsansätze zusammengetragen und besonderer Wert auf die Umsetzung von Ideen gelegt. Das Programmieren erlaubt und erzwingt es geradezu, seine Lösung durch praktische Tests zu überprüfen und die zugrunde liegende gedankliche Arbeit zu bestätigen; ein Vorteil, den andere strukturorientierte Fächer oft nicht haben. Die sofortige Rückmeldung beim Arbeiten am Computer gibt über den individuellen Erfolg ebenso Auskunft wie über die Leistung der Gruppe beim Zusammenfügen von Programmbausteinen.

In der Jahrgangsstufe 11 wird zunächst die selbständige und konstruktive Nutzung der vorgegebenen Programmierumgebung im Vordergrund stehen. Von Anfang an bearbeiten Schülerinnen und Schüler wachsende Anteile gestellter Aufgaben eigenständig: sie entwickeln Entwürfe zu Teilen einer (Gesamt)Lösung, formulieren umgangssprachlich begrenzte Beispiellösungen und übertragen sie in die Programmiersprache. Mit fortschreitendem Unterricht werden die zu leistenden Anteile immer mehr erweitert, bis die Schülerinnen und Schüler in der Lage sind, alle Arbeitsschritte zur Lösung einer informatischen Aufgabe selbst zu übernehmen.

2.3.2.3 Fortschreitende Analysetechnik und Bewertung

Mit zunehmenden Kenntnissen nimmt die Qualität von Beurteilungen zu. Sie bezieht sich auf die Analyse und Beurteilung des Lösungswegs, die Funktionalität eines entwickelten Programms und die Entwicklung von modifizierten Problemlösungen. Hierzu gehören auch Aspekte der theoretischen Informatik.

Welchen Anteil die Informatik an der Lösung des Ausgangsproblems hat, lässt sich mit wachsenden Fachkenntnissen von Mal zu Mal qualifizierter beurteilen. Unterschiedliche fachbezogene und fachübergreifende Anteile einer Problemlösung werden bewusst gemacht und kritisch bewertet. Die Ausgangssituation wird klar formuliert und nach Ausarbeitung der infor-

matischen Lösung noch einmal in die Gesamtbewertung mit einbezogen. Dabei werden die Lösung selbst und ihr Aufwand geprüft; zusätzlich ist von großer Bedeutung, wie die Lösung aus der Sicht der Auftraggeber und der Anwender eingeschätzt wird. Gesichtspunkte für alternative Lösungen lassen sich zu Beginn des Lösungsprozesses festhalten und in der Bewertungsphase wieder aufgreifen.

2.3.3 Darstellung der Obligatorik am Beispiel unterschiedlicher Sprachkonzepte

Der Unterricht hat sich an den fachlichen Inhalten im Sinne von Kapitel 2.1.1 zu orientieren und die Anwendungs- und Methodengesichtspunkte gemäß Kapitel 2.1.2 und 2.1.3 zu berücksichtigen. Bei der Anbindung an konkrete Gegenstände ist Freiraum gegeben, um auch zeitgemäße Entwicklungen berücksichtigen zu können.

Wie schon dargestellt, sind im Laufe der Zeit eine ganze Reihe von Informatiksprachen mit unterschiedlicher Präzision und Ausdruckskraft entstanden, die die Problemerfassung, das Denken und letztendlich die Problemlösungen entscheidend prägen. Damit sind die konkreten Unterrichtsinhalte zu einem Teil abhängig von der Paradigmenwahl, d.h. von der gewählten Alternative z. B. eines imperativen, objektorientierten, wissensbasierten oder funktionalen Ansatzes. Dies betrifft vor allem die fachlichen Inhalte für das in 2.1.1 beschriebene Feld „Modellieren und Konstruieren“. Weitgehend unabhängig vom gewählten Sprachenkonzept sind dagegen die fachlichen Inhalte im Feld „Analysieren und Bewerten“, die hier genannten Themen haben einen eigenen Stellenwert und werden an geeigneten Stellen in die Lernsequenzen (Kapitel 3) eingearbeitet. Für ein Unterrichtskonzept über eine Dauer von drei Jahren ist schließlich ein Paradigmenwechsel wünschenswert, damit die Schülerinnen und Schüler zumindest exemplarisch erfahren, dass bestimmte Problemklassen durch grundlegend andersartige Denkansätze mit neuen Werkzeugen vielfach effizienter gelöst werden können.

Es folgen nun sachlogisch geordnet und auf Kapitel 2.1 bezogene fachliche Inhalte für verschiedene Sprachkonzepte. Kursiv dargestellte Begriffe konkretisieren und ergänzen die obligatorischen fachlichen Inhalte aus der Sicht des Sprachparadigmas. Für das Feld „Analysieren und Bewerten“ sind jeweils nur die Aspekte benannt, die über die allgemeinen Inhalte hinaus besonders geeignete Zugänge aus Sicht des Sprachkonzeptes zulassen. Eine chronologische Reihenfolge für den konkreten Unterricht darf hieraus noch nicht abgeleitet werden, hierzu wird auf die Beschreibung der Lernsequenzen in Kapitel 3 verwiesen.

2.3.3.1 Imperativer Ansatz

Informatikmodelle gewinnen:
Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren

Bei der Entwicklung einer Problemlösung geht man von der Grundvorstellung aus, dass dem Computer als Subjekt in einem *Gesamtplan* eine *Abfolge von Handlungsanweisungen* zur Abarbeitung übergeben wird. Ausgehend von der Frage, welche *Ausgaben* aus welchen *Eingaben* erzeugt werden sollen, wird zielorientiert ermittelt, welche *Operationen* auf welchen *Datenobjekten* im Rahmen eines Gesamtprogramms nacheinander auszuführen sind, um den aktuellen Zustand der Objekte schrittweise zu verändern (*Top-Down-Methode*). Bei komplexen

Problemstellungen werden *strukturierte Lösungen* angestrebt, in denen die Erledigung von *Teilaufgaben* an Unterprogramme *delegiert* wird.

Daten und Algorithmen abstrahieren

Als typische Vorgehensweisen bei der Entwicklung einer Lösungsmethode zu einem spezifizierten Problem kommen *schrittweise Verfeinerungen, prozedurale Zerlegung und Datenabstraktion* zum Einsatz. Zunächst erweisen sich die von einer imperativen Entwicklungsumgebung bereitgestellten *Standarddatentypen, Grundoperationen, Strukturen zur Ablaufsteuerung und Modularisierungshilfen* bei einer Reihe von Aufgabenstellungen als hinreichende Formulierungshilfen. Komplexe Problemsituationen führen dann zu *spezifischen Datenorganisationen*, deren zugehörige *Typen und Operationen* von der Sprache nicht mehr bereitgestellt werden. Diese werden im Sinne einer *Spracherweiterung* als *Allgemeine Datentypen* wie *Listen, Dateien, Graphen* oder *Bäume* selbst definiert und anschließend auf der Basis verfügbarer Werkzeuge *implementiert*. Ergänzend dazu kann durch einen basisorientierten Ansatz zunächst geprüft werden, für welche Teilaufgaben bereits Lösungen bekannt sind oder leicht konstruiert werden können, um daraus dann die Gesamtlösung zusammenzusetzen (*Bottom-Up-Methode*).

Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

Orientiert am Ein- und Ausgabeverhalten des realisierten Programms ist die Lösung auf *Plausibilität* zu untersuchen und die generelle *Zweckmäßigkeit des Lösungsansatzes* und die *plangemäße Übersetzung* in die Programmiersprache zu überprüfen. Weiter gehende Untersuchungen beziehen sich auf die *Güte von Lösungen* (z. B. Laufzeitverhalten und Ressourcennutzung, übersichtliche Darstellung und Strukturierung, konsequente Schnittstellenverwendung, Angemessenheit der herangezogenen Standardalgorithmen).

Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen

Untersuchungen zum *Algorithmenbegriff*, zu den grundsätzlichen *Möglichkeiten und Grenzen der Algorithmisierbarkeit*, zur *Berechenbarkeit* und zur *Komplexität von Lösungen* sind historisch mit bewusst primitiven, imperativ geprägten Ansätzen verknüpft und können an Beispielen imperativer Programme gut untersucht werden.

Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen

Da viele Elemente imperativer Programmierung direkt auf der Grundlage der Konzeption einer *von-Neumann-Maschine* basieren, lässt sich der Prozess der *Transformation der Daten- und Ablaufstrukturen* von der höheren Programmiersprache zur *maschinennahen Sprache* organisch einleuchtend vollziehen. Die gewonnenen Erkenntnisse über den *funktionalen Aufbau*, über *Operationen und Adressierungsprinzipien* und die *Organisation des Speichers* auf der

maschinennahen Ebene vertiefen andererseits das Verständnis für Konstrukte in der imperativen Programmiersprache.

2.3.3.2 Objektorientierter Ansatz

Informatikmodelle gewinnen:
Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren

Problemsituationen können mit Hilfe *objektorientierter Analyse*verfahren modelliert werden. Diesem Ansatz liegt die Vorstellung zugrunde, dass *Objekte* selbständig Aufträge ausführen und interagieren können. Bei der Entwicklung einer Problemlösung geht man von dem Gedanken aus, dass die Realität stets durch Objekte beschrieben wird, die Informationen festhalten und miteinander kommunizieren können. Dabei steht der Objektbegriff im Mittelpunkt der Betrachtung. (Ein Objekt ist eine Einheit, die durch ihre *Attribute* und *Dienste* gekennzeichnet ist.)

Schon bei der ersten Problemeingrenzung wird mit Hilfe von Methoden der objektorientierten Analyse (OOA) das Finden und eine *Klassifizierung* der Objekte durchgeführt. Außerdem werden *Objektbeziehungen* beschrieben und mit geeigneten *Dokumentationstechniken* dargestellt.

Begleitend zur Analyse muss mit Hilfe von Veranschaulichungstechniken die Beziehung der Objekte visualisiert werden. Die Beschreibung der statischen Sicht der Objektbeziehungen wird durch *Objektdiagramme* beschrieben, die dynamische durch *Interaktionsdiagramme*. Das Ergebnis ist eine Welt von Objekten, die über Informationen verfügen, die ihre *Eigenschaften* und ihren aktuellen *Zustand* beschreiben und *Nachrichten* versenden.

Daten und Algorithmen abstrahieren

Beim *objektorientierten Design* (OOD) wird geprüft, ob und inwiefern *Klassen* aus vorhandenen *Klassenbibliotheken* benutzt werden können oder ob neue Klassen entworfen werden müssen. Typisch für die Objektorientierung ist die Verbindung von *Wiederverwendbarkeit* und Entwurf neuer Klassen durch das Konzept der *Vererbung*. Dabei entstehen aus bereits vorhandenen *Oberklassen* durch *Erweitern* und *Überschreiben* neue *Unterklassen*, deren Objekte auf dieselbe Nachricht unterschiedlich reagieren können (*Polymorphie*). Klassen werden aber nicht nur durch *Spezialisierung*, sondern auch durch *Generalisierung* entwickelt. Dabei entstehen *abstrakte* und spezielle Klassen; eigene Klassenbibliotheken werden entworfen.

Entscheidend sind auch hier wieder die Dokumentationstechniken. Die Schnittstellen von Klassen werden im *Protokoll* beschrieben. Hier finden sich Attribute, Dienste, Klassenbeziehungen (*Ist-Beziehung* der Vererbung), Objektbeziehungen (*Hat-Beziehung = Aggregation*, *Kennt-Beziehung = Assoziation*). Dienste werden als Verträge nach dem *Client-Server-Modell* mit Hilfe von Vor- und Nachbedingungen beschrieben. Wie Objekte ihre Aufgaben erledigen, bleibt nach außen verborgen (*Kapselung*).

Die *Benutzungsoberfläche* der Software wird entworfen.

Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

Bei der *objektorientierten Programmierung* (OOP) werden Programme als kooperierende Ansammlung von Objekten realisiert. Dabei werden die Attribute in Form von Zustandsvariablen und die Dienste als Methoden implementiert. Diese Implementationen werden in *Modulen* gespeichert, sodass unerlaubter Zugriff auf Interna von Objekten verhindert wird. Polymorphie wirkt sich so aus, dass durch *spätes Binden* erst zur Laufzeit entschieden wird, welche Methode aufgerufen wird. Ein typisches programmtechnisches Verfahren der OOP ist die *ereignisgesteuerte Programmierung*.

Durch die *Abstraktion* in OOA und OOD wird eine möglichst weit gehende Unabhängigkeit der Objekte erreicht, sodass einzelne Objekte oft bereits durch das Senden geeigneter Nachrichten getestet werden können. Schwieriger ist es, wenn vielfältige Abhängigkeiten zwischen den Objekten bestehen. In diesem Fall müssen zusammenhängende Teilgruppen von Objekten getestet werden.

Die Wiederverwendbarkeit von Klassen ermöglicht eine leichte Weiterentwicklung.

Typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen

Typische Anwendungen von OOP sind z. B. Animationen, grafische Benutzungsoberflächen, objektorientierte, leicht erweiterbare Betriebssysteme, Client-Server-Architekturen. Hier können Einsatzmöglichkeiten und Grenzen durch Vergleich mit kommerziellen Produkten beurteilt werden.

Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen

Weil wesentliche Konzepte der Automatentheorie (Zustände, Eingaben als Nachrichten) aus der OOP bekannt sind, lassen sich Automatenmodelle leicht mit Objekten realisieren. Mit Hilfe der Automaten können Compiler entwickelt und dabei Sprachkonzepte beurteilt werden.

Technische, funktionale und organische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen

Auf der Grundlage des Nachrichtenkonzepts der OOP sollte besonderes Gewicht auf Kommunikationsstrukturen gelegt werden. So können z. B. Busstrukturen mit Broadcasting beschrieben oder Protokolle als informationstechnische Grundlage des Austauschs von Nachrichten behandelt werden. Durch die objektorientierte Darstellung eines Computersystems kann die Funktionsweise der einzelnen Komponenten besser verstanden werden. Ebenso ist es auf diese Weise möglich, die grundlegenden Prinzipien aller Internetprotokolle zu erarbeiten.

2.3.3.3 Wissensbasierter Ansatz bei Verwendung einer PROLOG-Programmierumgebung

Informatikmodelle gewinnen:
Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren

Da für diesen Ansatz die Vorstellung charakteristisch ist, dass in einem Rechnersystem *Wissen* im Sinne der Beschreibung von Zuständen eines Ausschnittes der Welt sowie von *Methoden* zum Umgang mit diesem Wissen gespeichert ist, und dass dieses System in der Lage ist, *Folgerungen* aus diesem Wissen zu ziehen und zu *Fragen* eines Anwenders passende *Antworten* abzuleiten, werden reale Sachzusammenhänge so analysiert, vereinfacht und beschrieben, dass das Wissen darüber als *logisches Geflecht* zwischen den Objekten erfasst wird und *Relationen* zwischen den Objekten des Problemraumes aufgestellt werden.

Daten und Algorithmen abstrahieren

Eine Umsetzung in Richtung einer PROLOG-Programmierumgebung beschreibt das Wissen über die Objekte durch *Klauseln aus Fakten und Regeln.* Dazu werden Kenntnisse zur Definition von *Prädikaten und deren Stelligkeit, Wenn-Dann-Beziehungen, Konjunktionen, Disjunktionen,* die *Verteilung von Alternativen* auf Klauseln und Formulierungen von *Anfragen* erarbeitet und die Bedeutung von *Konstanten, freien und gebundenen Variablen* in Definitionen und Anfragen geklärt. *Rekursive Prädikatsdefinitionen* sind dabei frühzeitig einzubeziehen. Anspruchsvolle Problemlösungen kommen ohne die Datenstruktur *LISTE* und die zugehörigen *Listenoperationen* nicht aus. Zentral sind dabei die *Kopf-Rest-Methode*, die Veranschaulichung von Listen durch die zugrunde liegende *Baumstruktur* und die damit verbundene generelle *Konzeption von Datenstrukturen* in PROLOG-Systemen. Die Erarbeitung von typischen *Suchverfahren* im Zusammenhang mit Bäumen und Graphen ist notwendiges Hilfsmittel zur Lösung vieler Problemklassen.

Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

In PROLOG-Systemen muss zwischen logischem Gehalt und der Ablaufsteuerung streng unterschieden werden. Die Konzentration auf die Beschreibung des Problems, die Beschränkung auf die logischen Beziehungen zwischen Voraussetzung und Behauptung unterstreichen die *deklarative Bedeutung* eines logischen Programms. Die *prozedurale Bedeutung* ist durch den implementierten Algorithmus der Programmausführung bestimmt und abhängig von der zugrunde liegenden PROLOG-Maschine. Hier hängen Effizienz und Terminierung von der Reihenfolge der Programmklauseln und Ziele ab. Die in PROLOG-Systemen eingebauten Mechanismen zur *Unifikation, Resolution und automatischen Rückverfolgung* sind zu klären, in entsprechende Modelle wie *Ableitungsbaum* und *Vierportmodell* einzuführen und Möglichkeiten zur *Ablaufverfolgung* und *Ablaufsteuerung* zu nutzen, um Programme effizienter zu gestalten.

Typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen

Speziell *Datenbank- und Expertensysteme* werden in ihrem Aufbau und der Arbeitsweise diskutiert, analysiert, erläutert und bewertet. Eigene kleine Modelle werden entworfen und realisiert. Auf die Datenschutzproblematik wird an den Beispielen „Verknüpfung unterschiedlicher Datenquellen“ und „Rasterfandung“ eingegangen. Expertensysteme geben Anlass, über den Intelligenzbegriff, menschlicher versus maschineller Intelligenz und ethische Fragen nachzudenken.

Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen

Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen

Wie schon oben dargestellt, sind die Themen dieser beiden Unterpunkte weitgehend unabhängig vom gewählten programmiersprachlichen Ansatz. Genutzt werden muss bei einem prologbasierten Ansatz aber die ausgezeichnete Möglichkeit, Untersuchungen zur *maschinellen Sprachverarbeitung* von der Definition von Grammatiken, formaler Sprachen und Automaten bis zu entsprechenden Maschinenmodellen zum Testen und Erzeugen zu führen und die maschinelle Verarbeitung mit PROLOG zu demonstrieren. Parser, Interpreter und Übersetzer für beliebige kleine Sprachen können definiert, entwickelt und über ein PROLOG-System realisiert werden.

2.3.3.4 Funktionaler Ansatz

Informatikmodelle gewinnen:
Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren

Ein funktionaler Ansatz analysiert ein gegebenes Problem so, dass sich das gewünschte Verhalten des Systems als *Funktion* im Sinne der Mathematik in eventuell mehreren *Variablen* darstellt, deren Verhalten nur von den *Eingabedaten* abhängt. Eine zielgerichtete Analyse zerlegt komplexe Probleme in Teilprobleme, die jeweils eine eigene Funktion zur *Modellierung* benötigen. Man erhält eine Sammlung von Funktionen (*Umgebung*), die an Namen gebunden werden und definiertes Input-Output-Verhalten aufweisen.

Daten und Algorithmen abstrahieren

Rechenvorschriften, die das Verhalten der Funktion bestimmen, werden so weit zerlegt, bis sie sich aus bereitgestellten *Grundfunktionen (primitiven Ausdrücken)* zusammengesetzt zeigen. Die sich ergebenden Ausdrücke haben in der Regel Baumstruktur. Grundlegend ist der Mechanismus, aus gegebenen Funktionen schrittweise neue aufzubauen *(Kombination)* und diese an neue Namen binden zu können *(Abstraktion)*. Dabei sind eigene Funktionen genauso benutzbar wie vordefinierte, was den *Aufbau von Bibliotheken* ermöglicht. Zur Ablaufsteuerung benötigt man *logische Verknüpfungen* und *Fallunterscheidungen*.

Komplexere Daten werden als *Liste* auf der Grundlage des Konzepts der *Paarbildung* repräsentiert. Die *Kopf-Restliste-Methode* ist ausreichend, um beliebige allgemeine Datenstrukturen darzustellen. Zu ihnen zählen *Lineare Listen, Keller, Schlangen* und *allgemeine Bäume* und darauf definierte Operationen wie *Suchen* und *Sortieren.* Die Realisierung anspruchsvollerer Aufgaben erfordert gerade im Zusammenhang mit Listen die *Rekursion.* Dabei ist zwischen *linearer, indirekter* und *baumartiger* Rekursion zu unterscheiden.

Der *Mechanismus der Bindung von Namen an Werte* ist universell; er funktioniert für die Speicherung einer Zahl (einer konstanten Funktion) genauso wie für eine Funktion in mehreren Variablen. So wird die Unterscheidung zwischen Daten und Funktionen obsolet: *Daten* können als *Funktionen* (bzw. komplexe Ausdrücke) interpretiert werden, *Funktionen* als *Daten* in anderen Funktionen benutzt bzw. von ihnen ausgegeben werden.

Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

Zur Implementation einer Umgebung können die zu dieser gehörenden Funktionen unabhängig voneinander geschrieben und getestet werden, da das Verhalten einer Funktion nur von ihren Eingabedaten abhängt. Eine Lösung für ein konkretes Eingabedatum erhält man, indem man die Ausdrücke auswertet *(eval)* und dann die Funktion auf die Argumente anwendet *(apply)*. Dazu ist ein *Auswertungsschema* nötig, im einfachsten Fall ein *Substitutionsmodell*.

Die Darstellung „großer“ Systeme als Ansammlung von Objekten mit je individuellem Verhalten erfordert die Einführung *lokaler Variablen*, die den *Zustand* eines Objektes darstellen. Dies führt den reinen funktionalen Ansatz an eine Grenze, da nun *Wertzuweisungen* als *Seiteneffekte* unausweichlich werden. Gleichwohl lassen sich größere Systeme (z. B. eine arbeitende Bank) darstellen, wenn ein anderes Auswertungsschema - *lexical scoping* - herangezogen wird. Ähnlich lassen sich zeitliche Abläufe durch *Streams* und das Konzept der *verzögerten Auswertung* behandeln.

Typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen

Die Flexibilität der Listenverarbeitung begründet den Einsatz funktionaler Systeme in vielen Bereichen der „künstlichen Intelligenz“ (KI). Als Beispiele können Textbearbeitung und Dialogsysteme aber auch einfache Lösungssysteme für Probleme aus der symbolischen Mathematik (Termumformung, symbolisches Differenzieren) oder der Kryptologie gelten.

Dabei ermöglicht die *funktionale Modellierung* von Ausschnitten der Wirklichkeit zwar deren *Simulation*, jedoch muss diese notwendigerweise beschränkt bleiben, weil in die funktionale Beschreibung immer nur eine begrenzte Anzahl an Informationen einfließen kann, die zudem nie *an sich* sondern nur *nach Transformation in eine der Maschine zugängliche Form* vorliegen. Die Grenzen gerade im Bereich der „künstlichen Intelligenz“ können an den Fähigkeiten von Dialogsystemen (z. B. „Maschine als Psychiater“) oder an Übersetzungsprogrammen deutlich gemacht werden.

Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen

Die Aufhebung der Trennung zwischen Daten und Programmen, die in der theoretischen Informatik bei der *Kodierung von Turingmaschinen* im Rahmen der Untersuchung zur *Berechenbarkeit* und den *prinzipiellen Grenzen* von Maschinen relevant wird, ermöglicht neben der Analyse dieser Gesichtspunkte die Beschäftigung mit *Grammatiken, formalen Sprachen* und den sie jeweils *erkennenden Automaten* im Rahmen der Behandlung *maschineller Sprachverarbeitung*. Ebenso lassen sich *Parser* und *Compiler* für einfache Sprachen selbst entwickeln und implementieren.

Technische, funktionale und organisatorische Prinzipien von Hard- und Software-Systemen kennen lernen und einordnen

Über die weitgehend unabhängig vom jeweiligen programmiersprachlichen Ansatz zu behandelnden Themenbereiche ermöglicht das *Prinzip der Wertzuweisung* die Simulation digitaler Schaltungen, deren Verhalten funktional dargestellt wird. *Signalgesteuerte Prozesse* lassen sich dabei mit Hilfe von *Streams* nachbilden. Das etwa in SCHEME implementierte Konzept der *Engine* kann generell für alle Prozesse eingesetzt werden, die mit Zeitvorgaben arbeiten.

2.3.4 Zusammenfassung inhaltlicher und methodischer Kompetenzen

Das Schulfach Informatik leistet einen wesentlichen Beitrag zum Erziehungs- und Bildungsauftrag der Schule, da es sich mit den Grundlagen und Anwendungen der Informations- und Kommunikationstechniken beschäftigt und damit auf die Bewältigung zukünftiger Lebenssituationen in der Gesellschaft vorbereitet und zur Persönlichkeitsentwicklung des Einzelnen durch Förderung seiner Urteils- und Handlungsfähigkeit und seines verantwortungsbewussten Umgangs mit Information und Technik beiträgt:

Im Laufe der Kurssequenz im Fach Informatik haben die Schülerinnen und Schüler den Computer als ein Werkzeug erfahren und erkannt, welches menschliche Fähigkeiten ergänzt und erweitert, menschliche Arbeit durch Rechnerlösungen verändert und vielfältige Auswirkungen auf die Nutzer von Rechnersystemen und andere unmittelbar oder mittelbar Betroffene ausübt. Die Schülerinnen und Schüler

- kennen typische Verfahren zur Modellbildung in der Informatik und können diese anwenden und bewerten,
- haben einen Überblick über unterschiedliche Zugänge und Ansätze beim Problemlösungsprozess gewonnen und können Problemlösungen unter verschiedenen Gesichtspunkten bewerten und verbessern,
- haben bei der Umsetzung von Problemlösungen und im Rahmen von Projektarbeiten unmittelbare Erfahrungen gesammelt über die Veränderung der individuellen Arbeitsprozesse und der Teamarbeitsprozesse beim Einsatz des Werkzeugs Computer und können diese Erfahrung zum Lösen von Problemen und zur Strukturierung des Problemlösungsprozesses einsetzen,
- verfügen über theoretische Grundlagen der maschinellen Verarbeitung von Informationen, der bei der Problemformulierung und -lösung verwendeten formalen Beschreibungsmethoden und der Funktionsweise von Informatiksystemen,

- sind in der Lage, Möglichkeiten und Grenzen algorithmischer Verfahren zur Problemlösung zu erkennen und in Hinblick auf ihre Komplexität zu beurteilen,
- können die Auswirkungen der Informations- und Kommunikationstechnik auf die Lebens- und Arbeitswelt besser einschätzen und die historischen und gesellschaftlichen Voraussetzungen und Rahmenbedingungen des Einsatzes neuer Technologien sicherer einordnen und beurteilen.

3. Unterrichtsgestaltung/Lernorganisation

3.1 Grundsätze der Unterrichtsgestaltung

Es ist Aufgabe des Unterrichts, das im Bildungsauftrag genannte Hauptziel der gymnasialen Oberstufe realisieren zu helfen, auf Studium und Beruf vorzubereiten. Die Unterrichtsorganisation soll dazu beitragen, dass die Schülerinnen und Schüler auf der Grundlage einer vertieften allgemeinen Bildung

- eine wissenschaftspropädeutische Bildung erwerben
- und Hilfen zur persönlichen Entfaltung in sozialer Verantwortung erhalten

(vgl. Kapitel 1 „Aufgaben und Ziele der gymnasialen Oberstufe“).

Wesentliche Bezugspunkte sind die Dimensionen einer wissenschaftspropädeutischen Bildung, die im Richtlinientext mit

- dem Erwerb wissenschaftspropädeutischen Grundlagenwissens
- der Entwicklung von Prinzipien und Formen selbständigen Arbeitens
- der Entwicklung von wissenschaftlichen Verhaltensweisen
- der Ausbildung von Reflexions- und Urteilsfähigkeit

umschrieben werden.

Der Unterricht ist also so anzulegen, dass diese Ziele erreicht werden können. Die Prinzipien, denen hierbei gefolgt werden soll, sind im Kapitel 3 des Richtlinientextes „Prinzipien des Lernens und Lehrens in der gymnasialen Oberstufe“ beschrieben. Hierbei ist sicherzustellen, dass auf der einen Seite eine gut organisierte fachliche Wissensbasis erreicht wird. Dazu gehören Theorien, Fakten, Methoden- und Prozesswissen. Auf der anderen Seite muss eine Balance zwischen fachlichem Lernen und Lernen in sinnstiftendem Kontext hergestellt werden.

Zusammengefasst soll sich die Unterrichtsorganisation daran ausrichten, dass

- die individuelle Schülerpersönlichkeit mit ihren Vorerfahrungen, Möglichkeiten und Leistungsdispositionen im Blick ist
- Schülerinnen und Schüler aktiv lernen
- Schülerinnen und Schüler kooperativ lernen
- Vorwissen abgesichert, aufgegriffen und Lernfortschritt ermöglicht wird
- die Aufgabenstellungen komplex sind
- die Aufgabenstellungen auch auf Anwendung und Transfer ausgerichtet sind.

Fachliche Systematik, verbunden mit dialogischen, problembezogenen und fachübergreifenden Lernarrangements, sind die inhaltlichen Bezugspunkte für die Lernorganisation (vgl. Kapitel 3 „Prinzipien des Lernens und Lehrens in der gymnasialen Oberstufe“).

Die Unterrichtsgestaltung in der Informatik leitet sich neben diesen allgemeinen Kategorien aus dem Selbstverständnis des Faches und seiner in Kapitel 1.1 des Lehrplans Informatik dargestellten didaktischen Konzeption ab. Zusammen mit den in Kapitel 2.1.1 aufgeführten fachlichen Inhalten und dem Charakter des Faches als einer „Methodenwissenschaft“ realisiert der Informatikunterricht immanent die Forderung nach wissenschaftspropädeutischer Bildung. Die *Analyse* von Problemstellungen, das *selbständige Entwickeln* von Lösungskonzepten und die *Bewertung* von Lösungen sind Kategorien, die weit über den innerfachlichen Blickwinkel hinaus grundlegendes wissenschaftliches Arbeiten erfahren und erlernen lassen. Da die Unter-

richtsgestaltung ferner eine ständige Nähe zu dem „Werkzeug“ Computer in seiner jeweils aktuellen und zeitbedingten Ausprägung vermittelt, leistet die Informatik auch einen Beitrag zu der im Studium, in der Berufswelt und immer mehr auch im gesellschaftlichen Alltag allseits geforderten und notwendigen „Medienkompetenz“. Das Vermitteln von Fertigkeiten im Bereich gezielter Recherche und mediengerechter Präsentation gefundener und verarbeiteter Informationen gehört daher ebenfalls zu den Prinzipien der Unterrichtsgestaltung.

Auf der anderen Seite hat die Unterrichtsorganisation die Lernenden dort abzuholen, wo sie mit ihren unterschiedlichen individuellen Persönlichkeiten, Vorerfahrungen, Möglichkeiten und Leistungsdispositionen herkommen. Dies ist im Fach Informatik der gymnasialen Oberstufe schon deshalb von besonderer Bedeutung, weil der Unterrichtende nicht wie in anderen „klassischen“ Fächern von vergleichbaren Vorkenntnissen in der Lerngruppe ausgehen kann. Die Tatsache, dass in einem Kurs Neuanfänger neben solchen sitzen, die sich schon privat intensiv mit dem Computer auseinander gesetzt haben oder im Differenzierungsbereich der Jahrgangsstufen 9 und 10 Informatik belegt haben, erfordert ein hohes Maß an Sensibilität. Bei der hier zu Anfang meist notwendigen Binnendifferenzierung muss eine sinnvolle Unterrichtsgestaltung wesentlich auch das *kooperative Lernen* im Blick haben und fördern. So sind z. B. Schülerinnen und Schüler mit erheblichen Vorkenntnissen in ihrer sozialen Verantwortlichkeit gefordert, ihre eigenen Bedürfnisse nach fachlicher Weiterentwicklung für eine kurze Zeit zurückzustellen und stattdessen ihre Aufgabe auch in der Betreuung der anderen Schülerinnen und Schüler zu sehen. Schließlich nutzt es wenig, die Neuanfänger passiv mit den ihnen fehlenden Informationen zu überschütten. Vielmehr müssen sie *aktiv* das erlernen können, was den anderen bereits geläufig ist. Dies berührt bereits die Gestaltung der einzelnen Lernprozesse, wie sie im nächsten Abschnitt beschrieben wird.

3.2 Gestaltung der Lernprozesse

Der Begriff **Unterrichtsmethode** umfasst die Summe der Unterrichtsschritte, Arbeitsformen, Lehr- und Lernformen, mit deren Hilfe der Unterricht strukturiert wird. Die Unterrichtsmethoden und -organisationsformen sollen durch die in Kapitel 3.1 dargestellten Grundsätze geprägt sein.

Der Unterricht folgt einer Gesamtplanung, die schüler-, gegenstands- und methodenorientiert ist. Eine zu enge Steuerung des Lernprozesses ist ebenso zu vermeiden wie eine unstrukturierte Offenheit.

Schülerorientierung bedeutet, dass die Schülerinnen und Schüler die Möglichkeit haben, im Unterricht an ihrem eigenen Erfahrungs- und Lernstand anzuschließen und dem Leitbild des aktiven und selbständigen Arbeitens zu folgen.

Gegenstandsorientierung bedeutet, dass die vorgesehenen Unterrichtsinhalte in einem breiten Wissens- und Anwendungsbereich (vgl. Bereiche des Faches) in einer über die drei Jahre der gymnasialen Oberstufe laufenden Sequenz aufgebaut werden, dass Wissenszuwachs entsteht und vernetztes Wissen möglich wird.

Methodenorientierung bedeutet, dass die Schülerinnen und Schüler sich im Medium der Unterrichtsinhalte die geforderten fachlichen und fachübergreifenden Methoden und die notwendigen Arbeitshaltungen und -dispositionen aneignen.

Auf gängige Unterrichtsmethoden (z. B. Lehrervortrag, Unterrichtsgespräch) wird an dieser Stelle nicht eingegangen. Nachfolgend werden die Verknüpfung von Zielen, Inhalten und Unterrichtsmethoden, d.h. die Lernarrangements beschrieben, die geeignet sind, dem Leitbild des aktiven und selbständigen Lernens zu dienen und eine Vernetzung des Wissens zu ermöglichen. Die Formen eigenverantwortlichen Lernens und Arbeitens, die die Schülerinnen und Schüler aktiv tätig sein lassen, sind hier von besonderer Bedeutung.

3.2.1 Kriterien für die Auswahl von Unterrichtsinhalten

Der Unterricht in den Jahrgangsstufen 11 bis 13 wird sequentiell aufgebaut. Die fachlichen, fachübergreifenden und methodischen Ziele des Faches sollen am Ende der Jahrgangsstufe 13 erreicht sein. Folgende Kriterien können bei der Inhaltsauswahl hilfreich sein:

- Der Aufbau der fachlichen Inhalte darf nicht zu einer Stoffhäufung führen. Es gilt das Prinzip des Exemplarischen, das sich auf wesentliche, repräsentative und bedeutsame Fachinhalte beschränkt, die geeignet sind, übertragbare Kenntnisse und Fertigkeiten zu vermitteln.
- Die Auswahl der Unterrichtsinhalte soll so erfolgen, dass Vorwissen aktiviert werden kann. Lernzuwachs und Progression müssen deutlich werden.
- Die ausgewählten Inhalte sollen in fachlicher und fachübergreifender Hinsicht methodisch selbständiges Arbeiten ermöglichen und entsprechende Kompetenzen progressiv aufbauen und sichern.

Für die Aufteilung der Unterrichtsinhalte ist das schon in Kapitel 2.2.1 der Lehrpläne für das Fach Informatik beschriebene curricular spiralige Prinzip zu beachten, sodass schon in der Jahrgangsstufe 11 ein in sich geschlossener Durchgang durch wesentliche Bereiche der Informatik gewährleistet ist, der später vertieft wird. Dabei werden in jeder „Spiralwindung“ die Themen zunehmend komplexer, wobei auf Kontrastierung, Vergleich, immanente Wiederholung und zunehmende Theoriereflexion Wert zu legen ist.

Im Fach Informatik haben Rechnersysteme einen besonderen Stellenwert, ihr Einsatz unterscheidet sich qualitativ vom Stellenwert der Medien und Werkzeuge in den anderen Fächern. Einerseits ist die technische und logische Funktionsweise des gesamten Rechnersystems Gegenstand des Erkenntnisinteresses, andererseits ist es auch als Werkzeug zur Erkenntnisgewinnung notwendig. Durch entsprechende Software ist ein Rechnersystem an die unterschiedlichsten Anforderungen anpassbar. Deren Funktionalität verwischt die Grenzen zwischen der eigentlichen „Hardware“ Computer, deren lokaler physikalischer Vernetzung untereinander, einer globalen Vernetzung nach außen und den laufenden Betriebssystemen und Anwenderprogrammen. In ihrer Gesamtheit vereinigen Rechnersysteme typische Merkmale vieler klassischer Medien und erweitern diese um zusätzliche Komponenten, insbesondere um dynamische, interaktive Aspekte.

Die Konzeption, Ausstattung und Konfiguration des Gesamtsystems hat deshalb entscheidenden Einfluss auf die Auswahl von Unterrichtsinhalten und die Verwirklichung der Aufgaben und Ziele des Informatikunterrichts. Die Auswahl der Softwarewerkzeuge, angefangen bei den

Programmierwerkzeugen über gängige Standardsoftware bis zu spezieller didaktischer Software, ist eng an die angestrebten Unterrichtsziele (Paradigmenwahl) gekoppelt. Für Unterrichtssequenzen zur technischen Informatik können auch entsprechende Bausätze zum Messen, Steuern und Regeln notwendig sein. Zur Demonstration von Dokumentationen, Programmabläufen bzw. Schülerlösungen ist der Einsatz einer Bildschirmprojektionseinrichtung oder eines Videonetzes zwingend.

3.2.2 Lern- und Arbeitsorganisation im Fach

3.2.2.1 Konventionelle Arbeitsformen im Fach Informatik

Den konventionellen Lern- und Arbeitsformen wie z. B. Unterrichtsgespräch, Hausaufgaben, Klausuren, Referate und Protokolle ist im Fach Informatik der gleiche Stellenwert zuzuordnen wie in anderen Fächern, auf Besonderheiten wird in Kapitel 4.3.2 im Zusammenhang mit der Leistungsbewertung eingegangen.

Die besondere Bedeutung, die den Gruppen-, Team- und Projektarbeitsformen im Fach Informatik zukommt, ist in den vorangegangenen Abschnitten schon mehrfach dargestellt worden, insbesondere ihr Beitrag zur Entwicklung und Förderung des selbständigen Arbeitens ist in Kapitel 2.2.3 im Zusammenhang mit den fachlichen Inhalten und dem Lernen im Kontext erläutert. Derartige handlungs- und projektorientierte Arbeitsformen wird man in der Regel dann entwickeln und zum Tragen kommen lassen, wenn als Ziel die Erstellung eines Produktes gedacht ist. Dies kann ein Softwareprodukt (z. B. die Erstellung einer spezifischen Datenbank, eines Computerspiels, einer Seite im weltweiten Kommunikationsnetz etc.) sein oder auch ein hardwarenahes Produkt (z. B. die Steuerung einer Ampelanlage, die Vernetzung der Schulcomputer etc.) sein. Da ein solches Projekt zumeist recht umfangreich und komplex ist, kann es nur durch arbeitsteilige Gruppenarbeit gelöst werden. Dazu muss der Unterrichtsprozess detailliert organisiert werden und es sind fundierte Absprachen Voraussetzung für das Gelingen. Um die Ergebnisse arbeitsteiliger Gruppenarbeit oder eines Projektes allen Beteiligten verfügbar zu machen, damit der Einsatz des Produktes oder eine sinnvolle Weiterarbeit in Teilgruppen ermöglicht wird, kommt Protokollen und der Dokumentationen der Arbeit und ihrer Ergebnisse eine wichtige Funktion zu. Eine mediengerechte Aufbereitung und Verwaltung dieser Dokumente im Schulnetz, auf Datenträgern und in gedruckter Form erscheint sinnvoll und sollte daher frühzeitig eingeplant und vorbereitet werden.

Wie schon mehrfach dargestellt, ist die Modellbildung eine für die Schulinformatik wesentliche fachspezifische Methode, ganz gleich für welches Sprachparadigma man sich entschieden hat. Reale Objekte oder Objekte des Denkens müssen formalisiert, strukturiert und auf die Möglichkeiten der Maschine abgebildet werden. Unverzichtbares methodisches Hilfsmittel sind dabei die verschiedenen Techniken zur Modularisierung. Frühzeitig für den Unterricht nutzbar machen sollte man unterschiedliche, in der Regel mit einem Programmierparadigma verknüpfte Visualisierungsmethoden des Modualisierungsprozesses, da sie alle Phasen der Entwicklung im Unterricht anschaulich unterstützen und dokumentieren helfen. Weitere methodische Varianten zur Motivation und Veranschaulichung sind in der Regel mit den konkreten Problemstellungen eng verknüpft. So können z. B. Rollenspiele das Prinzip der Rekursion oder der Verkettung durch geeignete Verweistechniken bildhaft erläutern und „erleben“ lassen.

3.2.2.2 Facharbeit

Wissenschaftspropädeutisches Lernen zielt darauf ab, die Schülerinnen und Schüler mit den Prinzipien und Formen selbständigen Lernens vertraut zu machen. Facharbeiten sind hierzu besonders geeignet. Jede Schülerin bzw. jeder Schüler soll im Verlauf der Schullaufbahn eine Facharbeit anfertigen.

Facharbeiten können in der Regel in der Jahrgangsstufe 12 nach Festlegung durch die Schule je eine Klausur für den ganzen Kurs oder für einzelne Schülerinnen und Schüler ersetzen. Facharbeiten können auch im Rahmen der „sonstigen Mitarbeit“ bewertet werden. Eine Facharbeit hat den Schwierigkeitsgrad einer Klausur; sie soll einen Schriftumfang von 8 bis 12 Seiten (Maschinenschrift) nicht überschreiten.

Die methodischen Anforderungen an eine Facharbeit sind im Unterricht vorzubereiten. Unter Umständen ist es zweckmäßig, wenn diese Aufgabe nach Absprache in der Schule vom Fach Deutsch übernommen wird.

Im Fach Informatik bieten sich besondere Möglichkeiten, über Facharbeiten Themen aus größeren Sachzusammenhängen zu bearbeiten und sich mit dem konkreten Einsatz der Informations- und Kommunikationstechnologien in den verschiedensten Bereichen auseinander zu setzen. Es können fachspezifische, fachübergreifende oder fächerverbindende Themen bearbeitet werden. Die Festlegungen sind in den Fachkonferenzen abzustimmen.

In der Informatik eignen sich für die Facharbeiten übergeordnete Rahmenthemen. Es können auch Aufträge erteilt werden, die Besuche und Befragungen bei Firmen und Institutionen erforderlich machen und Einsatzbereiche und Probleme der Datenverarbeitung zum Thema haben. Programmierarbeiten beinhalten die Entwicklung eines Programms mit den aus dem Unterricht erwachsenen Werkzeugen und Methoden, mit allen Phasen vom Erfassen der Problemstellung bis zur Dokumentation. Die folgenden Vorschläge sollen Schlaglichter auf das Spektrum möglicher Themen werfen:

- Was verändert sich durch die neuen Informations- und Kommunikationstechnologien in der Stadtverwaltung?
- Eine Firma, Schule u.a. stellt sich vor. Wie erfolgt die Informationsaufbereitung, -darstellung und der Zugriff im Internet?
- Effizienzuntersuchungen am Beispiel verschiedener Lösungsalgorithmen zu einem Problemfeld.
- Untersuchungen zur „Gerechtigkeit“ von Mandatsverteilungsverfahren mit dem Rechner.
- Überlegungen zu einem Auskunfts- und Ausleihsystem zur Verwaltung der Schülerbücherei.
- Wie kann der Computer natürlichsprachliche Sätze verarbeiten? Analyse und Erzeugung natürlichsprachlicher Sätze.
- Strategiespiele - Warum ist der Computer kaum zu schlagen?
- Verschlüsselung von Nachrichten. Untersuchungen zu einigen Verfahren.
- Kreditkarten, Geldautomaten, Datensicherheit und Kryptologie. Wie sicher ist elektronisches Geld?
- Alternative Rechnerkonzepte durch neuronale Netze?

Eine Facharbeit hat den Schwierigkeitsgrad einer Klausur. Für die Ausarbeitung der Facharbeit im Fach Informatik kann im Vergleich zu einer Klausur erwartet werden, dass

- die Recherchen zum Thema zeitinteniver sind und unterschiedliche Medien, insbesondere das Rechnersystem und eventuelle Internetzugänge genutzt werden,
- der Beschreibungs- und Dokumentationsumfang größer ist und medienspezifische Präsentationsaspekte berücksichtigt,
- eventuelle Spezifikationen und Lösungsansätze selbständiger vorgenommen werden,
- eine Programmgestaltung in Bezug auf das Ein- und Ausgabeverhalten nutzungsorienterter bearbeitet wird,
- die Reflexionsebene eine höhere Betonung erfährt.

Da Schülerinnen und Schüler im Allgemeinen dazu neigen, sehr anspruchsvolle Programmieraufgaben und an kommerziellen Produkten orientierte Realisierungen vorzuschlagen, muss die Lehrerin bzw. der Lehrer hier korrigierend eingreifen und darauf bestehen, dass die Schülerinnen und Schüler unterrichtsgerechte Einschränkungen vornehmen. Die Dokumentation beginnt schon mit der Formulierung des Problems und der Spezifikation und muss alle weiteren Entscheidungsschritte enthalten. Die Gestaltung der Dokumentation und die Nutzung geeigneter Werkzeuge müssen vorher im Unterricht thematisiert werden. Entscheidend für die Facharbeit ist die vollständige Erledigung der gestellten Aufgabe in der vorgesehenen Zeit, auch wenn eventuell hinsichtlich der Perfektion der Lösung Abstriche zu machen sind. Eine Darstellung zur Reflexion auch des Arbeitsprozesses ist sinnvoll.

3.2.3 Fachübergreifende, fächerverbindende und projektorientierte Lern- und Arbeitsorganisation

Fachübergreifender Unterricht findet zunächst im Fach selbst statt; er besteht aus dem „Blick über den Tellerrand“ in Gestalt von Exkursen oder der Reflexion der fachlichen Fragestellung und ihrer Plausibilität und Grenzen.

Fächerverbindender Unterricht besteht in der themen- oder problembezogenen Kooperation zweier oder mehrerer Fächer, wenn es gilt, „quer liegende“ Themenstellungen unter verschiedenen Fachperspektiven und -kategorien zu betrachten und dabei mehr als nur die Summe von Teilen zu erkennen. Fächerverbindender Unterricht ist organisatorisch und planerisch aufwendig. Er kann in den Schwerpunkten eines Schulprofils entwickelt werden. Da die Schülerinnen und Schüler in der gymnasialen Oberstufe an **einer** übergreifenden Veranstaltung teilnehmen sollen, müssen die Schulen, sofern sie keine Schulprofile (Fächerkopplungen) aufweisen, entsprechend langfristig planen.

Projektorientierter Unterricht ist anwendungsbezogen, kurzphasig, kompakt, produktorientiert. Er muss in der Themenstellung erkennbar „besonders“ und machbar sein. Er kann im Fach selbst oder fächerverbindend stattfinden.

Fächerverbindender Projektunterricht findet in **übergreifenden Projektveranstaltungen** statt. Diese Veranstaltungsform soll den Schülerinnen und Schülern die Möglichkeit geben, erlernte Arbeitsmethoden aus unterschiedlichen Fachbereichen selbständig auf ein komplexes Problem zu beziehen und ein Problem aus der Perspektive mehrerer Fächer zu sehen. Projekt-

veranstaltungen bieten auch die Gelegenheit zur Teamarbeit. Diese Veranstaltungen sind unter bestimmten vorher festgelegten Leitfragen langfristig aus dem Fachunterricht heraus zu entwickeln. Die von den Schülerinnen und Schülern erbrachten Leistungen werden im Rahmen der „sonstigen Mitarbeit“ beurteilt.

Da solche Projektveranstaltungen stufenspezifische Ziele verfolgen, sind sie im Hinblick auf die Teilnehmerinnen und Teilnehmer in der Regel auf eine Jahrgangsstufe oder auf die gymnasiale Oberstufe zu beschränken.

Fachübergreifend sind die Methoden der Informatik in aller Regel dann, wenn es um Strategien zu Problemlösungen in Anwendungsbereichen geht. Hier werden Fragestellungen aus anderen Fachgebieten analysiert, spezifische Methoden der entsprechenden Fächer auf ihre Effizienz überprüft und mit bekannten oder neu entwickelten Methoden informatorischen Charakters in Verbindung gebracht. Da die Problemstellungen für die Informatik fast immer von außerhalb kommen, ist diese Situation eher der Normalfall. Natürlich sollten die Gegenstände aus den anderen Fachbereichen hinreichend bekannt sein, damit sich die Einarbeitungszeit in Grenzen hält. Weiter gehende Fragestellungen sind nur in Projektveranstaltungen möglich.

Das Fach Informatik kann hierzu naturgemäß eine Fülle von Fragestellungen beitragen. Dabei ist hier weniger an die üblichen Hilfestellungen gedacht, die bei jeder schulischen Projektveranstaltung von den Informatikerinnen und Informatikern erwartet werden (z. B. das Layout zur Projektzeitung, visuelle Aufbereitungen von Datenmaterialien oder Recherchen aus dem Internet), als vielmehr an Fragestellungen, zu denen die Informatik auch aus inhaltlich und methodisch eigenständiger Sicht Beiträge oder Antworten geben kann. Ohne jeden Anspruch auf Vollständigkeit können hier nur einige wenige Beispiele und Berührungspunkte stichwortartig genannt werden:

- Informatik und Intelligenzbegriff,
- Möglichkeiten der Informatik zur Beschreibung „dynamischer Systeme“,
- Einsatz informatischer Methoden in der Linguistik/Literatur,
- Einfluss der Informatik auf den „homo ludens“/„homo faber“,
- Bedeutung der Informatik für die „Informationsgesellschaft“,
- Überschneidungen zum Bereich der Kunst,
- Überschneidungen mit Aspekten der Musik.

3.2.4 Besondere Lern- und Arbeitsformen

Mit der „besonderen Lernleistung“ sollen herausgehobene Leistungen, die Schülerinnen und Schüler zusätzlich erbracht haben, im Rahmen der für die Abiturprüfung vorgesehenen Punktzahlen auch zusätzlich honoriert werden. Es muss sich um eine herausragende Leistung handeln. Dies hat auch in Art und Umfang der Darstellung bzw. der Dokumentation seinen Niederschlag zu finden. Die Kultusministerkonferenz hat als äußerliche Anhaltspunkte für die Wertigkeit den Rahmen bzw. den Umfang eines mindestens zweisemestrigen Kurses - dieses entspricht dem Äquivalent von maximal 60 Punkten - genannt.

Besondere Lernleistung kann z. B. sein: Ein umfassender Beitrag aus einem von den Ländern geförderten Wettbewerb, es kann das Ergebnis eines über mindestens ein Jahr laufenden fachlichen oder fachübergreifenden Projektes sein. Es kann sich auch um eine größere Arbeit han-

deln, die sich aus dem Fachunterricht ergeben hat. Die besondere Lernleistung muss in Qualität und Umfang eine Facharbeit deutlich überschreiten. Sie soll außer- und innerschulische Möglichkeiten außerhalb der Unterrichtsvorhaben erschließen, etwa in Feldarbeit und Experiment, in der Arbeit in Archiven oder Bibliotheken. Das Vorhaben soll eine klare Aufgabenstellung haben, eine nachvollziehbare Ausführungsebene (z. B. Produkt, Recherche, Versuch, Auswertung bzw. Reflexion).

3.3 Grund- und Leistungskurse

Grund- und Leistungskurse tragen gleichermaßen dazu bei, das Ziel der Studierfähigkeit zu erreichen.

Grundkurse repräsentieren das Lernniveau der gymnasialen Oberstufe unter dem Aspekt einer grundlegenden wissenschaftspropädeutischen Ausbildung.

Sie sollen
- in grundlegende Fragestellungen, Sachverhalte, Problemkomplexe, Strukturen und Darstellungsformen eines Faches einführen
- wesentliche Arbeitsmethoden des Faches vermitteln, bewusst und erfahrbar machen
- Zusammenhänge im Fach und über dessen Grenzen hinaus in exemplarischer Form erkennbar werden lassen.

Leistungskurse repräsentieren das Lernniveau der gymnasialen Oberstufe unter dem Aspekt einer exemplarisch vertieften wissenschaftspropädeutischen Ausbildung.

Sie sind gerichtet
- auf eine systematische Beschäftigung mit wesentlichen, die Komplexität und den Aspektreichtum des Faches verdeutlichenden Inhalten, Theorien und Modellen
- auf eine vertiefte Beherrschung der fachlichen Arbeitsmittel und -methoden, ihre selbständige Anwendung und theoretische Reflexion
- auf eine reflektierte Standortbestimmung des Faches im Rahmen einer breit angelegten Allgemeinbildung und im fachübergreifenden Zusammenhang.

Beide Kursarten basieren unverzichtbar auf dem Grundkursunterricht der Jahrgangsstufe 11.

Die Aufgaben und Ziele des Faches Informatik, die Bereiche des Faches in den Ausprägungen der fachlichen Inhalte, des Lernens im Kontext und der Methoden und Formen selbständigen Arbeitens sind für beide Kursarten gleichermaßen verbindlich.

Die Auswahl der fachlichen Inhalte muss für eine Grundkurssequenz weiter eingeschränkt werden, als dies für eine Leistungskurssequenz erforderlich wäre. Diese Einschränkungen betreffen das Feld „Analysieren und Bewerten“ etwas stärker als das Feld „Modellieren und Konstruieren“. In den Unterfeldern „Ein Informatikmodell gewinnen“ und „Typische Einsatzbereiche ...“ wird man im Rahmen einer Grundkurssequenz gegenüber einer Leistungskurssequenz insbesondere Einschränkungen hinsichtlich der Komplexität der zu behandelnden Problemfelder und Anwendungsbereiche vornehmen. Beim Prozess der Modellbildung wird das

ursprüngliche Problem weiter gehenderen Reduktionen zu unterwerfen sein als im Rahmen einer Behandlung im Leistungskurs.

In den Unterfeldern „Daten und Algorithmen abstrahieren“ und „Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln“ muss der Grundkurs die Anzahl der zu behandelnden Datenstrukturen und algorithmischen Strategien auf grundsätzliche einschränken, während im Rahmen einer Leistungskurssequenz hier durchaus ein systematischer Überblick anzustreben ist. Während in einer Leistungskurssequenz ausgehend von der Modellbildung der weitere Prozess bis zur Repräsentation und Implementation in einer Programmiersprache häufiger vollständig durchlaufen wird, kann der Grundkurs dies nur exemplarisch tun und muss nach einer funktionalen Spezifikation vielfach auf entsprechende vorhandene Werkzeuge einer Programmierumgebung zurückgreifen und diese zur Problemlösung einsetzen. Auch die Behandlung eines weiteren Programmiersprachenparadigmas wird man im Grundkurs nur exemplarisch durchführen können.

Aus dem Unterfeld „Algorithmen, Sprachkonzepte und Automatenmodelle beurteilen“ werden in einer Grundkurssequenz typische Automatenmodelle und geeignete formale Beschreibungssysteme exemplarisch zur Darstellung an verschiedenen Stellen im Modellierungsprozess herangezogen, während der Leistungskurs diese Werkzeuge systematischer und breiter einsetzt und gegebenenfalls auch im Anwendungskontext vergleichend analysiert. Auf eine Vertiefung im Bereich der theoretischen Informatik muss im Grundkurs in der Regel verzichtet werden.

Im Unterfeld „Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen“ wird in einer Grundkurssequenz der Schwerpunkt auf die exemplarische Behandlung einer typischen Architektur wie der von-Neumann-Maschine und der Transformation weniger Sprachelemente auf ein elementares Maschinensprachniveau gelegt. Der Leistungskurs wird sich auch mit alternativen Konzepten auseinander setzen. Für den Leistungskurs ist die Entwicklung eigener Simulationsumgebungen zur vertiefenden Behandlung der maschinellen Sprachverarbeitung denkbar, der Grundkurs wird auf entsprechende fertige Modellsysteme zurückgreifen.

Die Einführung in fachspezifische Methoden, insbesondere die Einübung in fundamentale Arbeitstechniken und Methoden, ist die Aufgabe sowohl von Grund- als auch von Leistungskursen. Die Behandlung spezieller Methoden, Methodenreflexion und Methodenvergleich wird eher dem Leistungskurs vorbehalten bleiben, weil hier eine breitere Entfaltung verschiedenartiger Konzepte möglich ist.

In beiden Kursarten muss den Schülerinnen und Schülern bewusst werden, dass das Fach Informatik in der gymnasialen Oberstufe in das enge gegenseitige Bedingungs- und Beziehungsgefüge der Fächer des mathematisch-naturwissenschaftlichen-technischen Aufgabenfeldes eingebettet ist, dass aber dennoch grundlegende Unterschiede zwischen der traditionell analytischen Denkweise der Naturwissenschaften und der konstruktiv-synthetischen Denkweise des Faches Informatik bestehen.

3.4 Sequenzbildung

In den folgenden Abschnitten sind Beispiele für Lernsequenzen dargestellt. Hierbei wird der curriculare Auftrag der Jahrgangsstufe 11 besonders berücksichtigt: Eingangskenntnisse angleichen, fachliche und fachübergreifende Fragestellungen kennen lernen und die Voraussetzungen für künftige Anforderungen der in Jahrgang 12 fortgeführten Grund- und Leistungskurse schaffen. Thematisch und fachmethodisch soll die Einführungsphase in der Jahrgangsstufe 11 einen in sich geschlossenen Durchgang durch wesentliche Teile der Schulinformatik vermitteln. Der Unterricht der Jahrgangsstufen 12 und 13 greift die Themen wieder auf, erweitert und vertieft sie nach dem Spiralprinzip.

Die Sequenzen werden in tabellarischer Form dargestellt, wobei die einzelnen Unterrichtsabschnitte durch ihre inhaltlichen Schwerpunkte und durchschnittliche Unterrichtszeiten charakterisiert werden. In der rechten Spalte wird ein Bezug zur Obligatorik hergestellt, der sowohl die allgemeine Obligatorik (Kapitel 2.3) als auch die paradigmenspezifischen Ausprägungen (Kapitel 2.3.3) berücksichtigt. Im Folgenden werden verschiedene Sequenzen aufgeführt, die sich an unterschiedlichen Sprachparadigmen orientieren.

3.4.1 Sequenzen für die Jahrgangsstufe 11

Die Aufgabe der Jahrgangsstufe 11 in ihrer allgemeinen Funktion ist im Kapitel 4 des Richtlinientextes beschrieben. Die Schülerinnen und Schüler belegen in der Jahrgangsstufe 11 i.d.R. durchgehend 10 bis 11 Grundkurse (30 bis 33 Wochenstunden). Der Unterricht folgt für die Jahrgangsstufen 11 bis 13 insgesamt einem Sequentialitätsprinzip. Dabei ergibt sich für die Jahrgangsstufe 11, dass sie die wissenschaftspropädeutische Vorbereitung für die Qualifikationsphase inhaltlich und methodisch übernehmen muss, d.h. dass gesorgt werden muss

- für eine breite fachliche Grundlegung,
- für eine systematische Methodenschulung in fachlicher, fachübergreifender und kooperativer Hinsicht,
- für Einblicke in die Anforderungen von Leistungskursen
- für Angebote zur Angleichung der Kenntnisse.

Die vorgestellten Sequenzen haben je nach Wahl des im Unterricht genutzten Sprachkonzeptes unterschiedliche Schwerpunkte und Ausprägungen. Sie besitzen exemplarischen Charakter und zeigen das Typische eines unterrichtlichen Hauptstrangs. Jedoch werden nicht alle in Abschnitt 2.3.3 angesprochenen Bereiche in jedem Fall und in allen Einzelheiten abgedeckt, hier verbleiben Gestaltungsspielräume.

(1) Sequenz „imperativ“

Sequenz für die Jahrgangsstufe 11 mit imperativem Ansatz

Für den imperativen Ansatz haben sich PASCAL-verwandte Sprachen nach Nikolaus Wirth im Unterricht der Sekundarstufe II durchgesetzt. Nutzt man über diese allgemeine Systemplattform hinaus die weiterentwickelten visuellen Programmierumgebungen, so besteht die Möglichkeit, bei der Erstellung eigener Programme ihre Hilfsmittel zur komfortablen Gestal-

tung von Benutzeroberflächen einzusetzen, ohne den etwa mit enthaltenen objektorientierten Charakter dieser Gestaltungswerkzeuge zu thematisieren. Zentral bleibt jedoch das EVA-Prinzip, wobei die Ein- und Ausgaben sowie das Programmlayout vom Gestaltungswerkzeug unterstützt werden. Die Verarbeitung erfolgt weiterhin im Rahmen der klassisch-imperativen Algorithmik.

Lernsequenz	Verknüpfung mit der Obligatorik
Allgemeine Einführung (2-4 Std.) Ausgehend von den Vorerfahrungen und Erwartungen der Schülerinnen und Schüler werden die Ziele des Unterrichtsfaches formuliert. Unterschiedliche Vorkenntnisse werden angeglichen, sodass auf der Basis der vorhandenen Hard- und Software ein einheitlicher Einstieg möglich wird. Dies schließt eine vorläufige Erklärung der Komponenten einer DV-Anlage und eine Einführung in ihre Bedienung ein.	
Einführung in die Programmierumgebung (15-21 Std.) Verwendet man einen einfachen PASCAL-Dialekt, so erscheint die Einführung anhand eines *datenfreien Programmiermodells* (z. B. Robotersystem NIKI) sinnvoll. Hierbei steht die *modularisierende* Arbeitsweise nach dem *Prozedurkonzept* im Vordergrund. Der Umgang mit Datenobjekten wird dann erst mit der nächsten Unterrichtsphase verknüpft.	Modularisierung und schrittweise Verfeinerung
Nutzt man dagegen eine visuelle Programmierumgebung, so stehen das *Variablenkonzept* und *einfache Datenstrukturen* am Anfang.	Einführung einfacher Daten- und Kontrollstrukturen
Anwendungsbeispiele für diesen Ansatz sind das Erstellen einer Telefonrechnung, Simulation eines Taschenrechners oder eines Spielautomaten usw. Inhaltliches Gerüst für diese Alternative bilden einfache Datentypen, Variablen, Zuweisungen, Kontrollstrukturen und das Prozedurkonzept.	Erste kleinere Anwendungsprobleme
Einfache Algorithmen und Datenstrukturen (30-36 Std.) Hier werden z. B. einfache mathematische Algorithmen, Such- und Sortierverfahren oder Textbearbeitungsprobleme in verbaler Form modularisierend erarbeitet und anschließend implementiert. Neben *exemplarischen Grundalgorithmen* bilden die Schnittstellenbeschreibung und die Technik der Kommunikation zwischen Programmteilen (*Parameterübergabe bei Prozeduren und Funktionen*) den Schwerpunkt. Die Reihenfolge der Bearbeitung richtet sich nach dem zuvor eingeschlagenen Weg.	Abstraktion einfacher Daten und Algorithmen, Probleme eingrenzen und spezifizieren Methodisch wechseln hier selbständige Arbeitsformen mit lehrergesteuertem Unterricht ab.
Aufbau und Arbeitsweise eines Computers (12-15 Std.) Es werden *grundlegende Ideen und Konzepte* herausgestellt (z. B. das *von-Neumann-Konzept*) und typische Grundeinheiten heutiger Computer in ihrer Wirkungsweise erläutert (z. B. Prozessor, Arbeitsspeicher, Bussystem) sowie in ihrem Zusammenwirken mit der Software beleuchtet (z. B. die Aufgaben eines *Betriebssystems*). Weiterhin wird an einem Beispiel aufgezeigt, welche Aufgaben ein *Compiler* prinzipiell zu leisten hat, d.h. in welchen Schritten eine in einer höheren Programmiersprache geschriebene Anweisungsfolge über mehrere Zwischenstufen in eine von der Maschine ausführbare Form übersetzt wird.	Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen
Komplexere Algorithmen mit allgemeinen Datentypen **(ADT)** (24-30 Std.) Von typischen *Anwendungen* ausgehend, z. B. einem Warteschlangenproblem, werden die logischen Strukturen herausgearbeitet (z. B. die Datenstruktur Schlange) und die Probleme mit Hilfe bereitgestellter Werkzeuge *(z. B. ADT Schlange)* programmtechnisch gelöst. Der Nutzung komplexerer Werkzeuge entspricht eine *projektbezogene arbeitsteilige* Vorgehensweise, die in der gemeinsamen Beurteilungsphase auch die Erörterung *gesellschaftsrelevanter Fragen* einschließt.	Realisierung, Überprüfung und Weiterentwicklung von Lösungskonzepten. Neben selbständigem Lernen treten hier zunehmend Aspekte kooperativen Lernens in den Vordergrund.

(2) Sequenz „objektorientiert allgemein“

Sequenz für die Jahrgangsstufe 11 mit objektorientiertem Ansatz unter Verwendung plattformunabhängiger Klassenbibliotheken

Lernsequenz	Verknüpfung mit der Obligatorik
Malen mit der Maus (12-17 Std.) **(Benutzen gegebener Klassen, das Nachrichtenkonzept)** Ausgangspunkt für die ersten Anwendungsentwicklungen ist eine einfache *Klassenbibliothek*, die wichtige Komponenten und die Grundfunktionalität eines Computersystems in anschaulicher Weise modelliert. Sie bleibt im nachfolgenden Unterricht als Basis einer dann erweiterten Klassenbibliothek erhalten.	Programmierkonzepte verstehen und benutzen
Die Basisbibliothek enthält die Klassen Maus, Tastatur, Bildschirm, die die entsprechenden Rechnerkomponenten modellieren, sowie die Klasse Stift, die dazu dient, einfache Zeichenoperationen auf dem Bildschirm durchführen zu können. In den Einstiegsprojekten werden ausschließlich Objekte dieser vier Klassen benutzt. Mit Hilfe der von ihnen angebotenen Dienste lassen sich sowohl anwendungs- als auch benutzergesteuert Zeichnungen erstellen. Hier wird der Umgang mit Objekten *(Nachrichtenkonzept)* anschaulich behandelt und es werden die wichtigsten *Kontrollstrukturen* vermittelt.	Verwenden einer Klassenbibliothek Algorithmen abstrahieren Lösungskonzepte implementieren und testen Das Konzept „Objekt“ verstehen und benutzen
Animationen (25-30 Std.) **(Klassenbildung, Vererbung durch Spezialisierung)** Typisch für Anwendungsprogrammierung nach dem objektorientierten Ansatz ist es, in einer gegebenen Problemsituation Objekte und ihre Beziehungen zu identifizieren *(Objektorientierte Analyse)*, sie in geeigneten Klassen abzubilden *(Objektorientiertes Design)*, und diese schließlich in der gewählten Programmierumgebung zu implementieren *(Objektorientierte Programmierung).*	
Zum Einstieg in diese Vorgehensweise sind Beispiele erforderlich, die eine Objektidentifikation möglichst unmittelbar zulassen. Hierzu eignen sich in hervorragender Weise auf dem Bildschirm animierte grafische „Objekte“ wie Windmühlen mit drehenden Flügeln, an den Rändern der Zeichenebene abprallende Kugeln, auf die Maus weisende Pfeile. Die Analyse führt zur exakten Festlegung von Attributen und Diensten der Objekte der beteiligten Klassen, die in *Klassenprotokollen* festgehalten werden. Attribute und Dienste werden mit Hilfe von Zustandsvariablen und Methoden gemäß der vorgegebenen Spezifikation implementiert.	Problembezogene Objekte spezifizieren, Klassenentwurf dokumentieren, Modularisierung; Objektorientierte Analyse Ein informatisches Modell gewinnen, Objektdiagramme zum Strukturieren, Objektbeziehungen

Die Auslagerung der Klassenimplementierungen in Module und die geforderte Eigenständigkeit der Klassen machen es notwendig, den *Objektbeziehungen* besondere Aufmerksamkeit zu schenken. Grafische Objekte beispielsweise besitzen in der Regel einen Stift (Hat-Beziehung); Bälle, die vom Bildschirmrand abprallen, müssen den Bildschirm kennen, um seine Dimensionen erfragen zu können (Kennt-Beziehung). Die Beschreibung der Objektbeziehungen erfolgt im Rahmen einer detaillierteren objektorientierten Analyse. Mit Hilfe geeigneter grafischer Darstellungsverfahren werden die Ergebnisse der Beziehungsanalysen anschaulich dokumentiert.	Client-Server-Modell, Nutzen fortschreitender Analysetechniken Aufdecken der Funktionsweise vorhandener Werkzeuge, Programmierkonzepte verstehen und benutzen
Die Wiederverwendbarkeit und Erweiterbarkeit von Klassen durch *Vererbung* ist ein weiteres zentrales Thema dieses Unterrichtsabschnittes. Werden Objekte mit zusätzlicher oder veränderter Funktionalität versehen, so lassen sich die zugehörigen Klassen vielfach durch Spezialisierung bereits vorhandener Klassen gewinnen (farbige Kugeln, Windmühlen mit verschiedenen Drehrichtungen)	Klassenbeziehungen, Selbstständiges Arbeiten: Entwicklung neuer Klassen, erste arbeitsteilige Gruppenarbeit
Geometrische Objekte für ein Grafikprogramm (17-23 Std.) **(Ereignissteuerung, Listen, Vererbung durch Generalisierung)**	
Typisch für aktuelle Anwendungen und Benutzungsoberflächen ist, dass ihr Ablauf durch Ereignisse wie Mausklicks oder Tastendrucke gesteuert wird. Diese Anwendungen stellen dem Benutzer häufig standardisierte Elemente wie Knöpfe oder Schieberegler zur Ablaufsteuerung zur Verfügung. Im Rahmen der Unterrichtssequenz werden die *charakteristischen Merkmale einer Ereignissteuerung* behandelt: In einer zentrale Ereignisschleife werden zu den jeweils aufgetretenen Ereignissen die entsprechenden Methoden (*Handler*) aufgerufen.	Benutzerführung untersuchen und bewerten, Ereignisgesteuerte Programmierung, Entwicklung neuer Werkzeuge, Projektorientiertes Arbeiten: Entwicklung verschiedener Klassen für Grafikobjekte
Die Grundlagen einer Ereignissteuerung werden anhand eines Grafikprogrammes erarbeitet: Grafische Objekte werden durch Mausdruck aktiviert (Prinzip des Knopfes) und mit Hilfe der Maus oder Tastatur bewegt, gedreht oder vergrößert. Neue Objekte können hinzugefügt, vorhandene entfernt werden.	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren
Dabei tritt ein weiterer Aspekt der Vererbung auf: *Generalisierung*. Unterschiedliche grafische Objekte ähneln sich in einer Vielzahl von Eigenschaften. Dies nutzt man aus, um eine (gegebenenfalls abstrakte) Oberklasse zu bilden, die gemeinsame Eigenschaften bündelt und aus der die gewünschten Klassen abgeleitet werden können (Klassenbildung durch Generalisiserung).	Klassenbeziehungen, Polymorphie, Entwicklung einer Datenorganisation, spätes Binden
Weil zur Verwaltung der grafischen Objekte eine dynamische Struktur erforderlich ist, bietet sich die Möglichkeit an, auf *Verkettungsstrukturen* als Vorläufer eines abstrakten Datentyps Liste einzugehen. Das erstellte Programm wird mit kommerziellen Programmen verglichen.	Einsatz von Informationssystemen untersuchen und bewerten Lernen im Kontext, Grafische Anwendungen
Getränkeautomat (22-27 Std.) **(Theoretische Informatik, Automatentheorie)**	
Ausgangspunkt für einen Einstieg in die theoretische Informatik könnte die Analyse eines Getränkeautomaten und seine Simulation im objektorientierten Modell sein. Das objektorientierte Modell beschreibt den Automaten durch seine Zustände und seine Zustandsübergänge. Die Analyse der Fähigkeiten dieses Automaten und strukturell ähnlicher Automaten führt zum *Begriff des endlichen determinierten Automaten*. Derartige Automaten können unter vielfältigen Aspekten betrachtet werden. Insbesondere können ihre Leistungsfähigkeit (etwa als Parser für reguläre Sprachen) und ihre Grenzen untersucht werden. In diesem Zusammenhang kann verdeutlicht werden, dass ein zentraler Gegenstand der Informatik die Erforschung der prinzipiellen Möglichkeiten und Grenzen eines Rechners mit Hilfe passender Automatenmodelle ist. Hier werden die Grundlagen gelegt, die im weiteren Verlauf des Informatikunterrichts zur Vorstellung eines allgemeinen Rechners (Turing-Maschine) und zum Begriff der Berechenbarkeit führen.	Lernen im Kontext: Getränkeautomaten, Computersimulation Objektorientierte Analyse, Analyse von Anwendungssituationen Ein informatisches Modell gewinnen, Kommunikationsstrukturen einordnen Automatenmodelle beurteilen Benutzerführung und Funktionsumfang untersuchen und bewerten

(3) Sequenz „objektorientiert visuell“

Sequenz für die Jahrgangsstufe 11 mit objektorientiertem Ansatz unter Verwendung visueller Werkzeuge

Lernsequenzen	Verknüpfungen mit der Obligatorik
Einführung in die visuelle und ereignisorientierte Programmierung (20-25 Std.)	
In Anlehnung an die wesentlichen Merkmale graphischer Benutzerschnittstellen wird eine erste, vorläufige Charakterisierung von „Programmierung“ erarbeitet. Programmieren bedeutet, das Ausehen, die Bestandteile und den Ablauf eines Computer-Programms festzulegen: die *Benutzeroberfläche* (Fenster, Schaltflächen, ...), die Auswahl der *Ereignisse* (Klicken, Tastendruck, ...) und die *Reaktionen des Programms* (= Befehle für den Computer).	Benutzerführung untersuchen, Problemstellung eingrenzen
Hierfür stehen vordefinierte Klassen in einer Klassenbibliothek zur Verfügung. Objekte dieser Klassen werden an die konkrete Problemstellung angepasst, indem ihr äußeres Erscheinungsbild (Position, Größe, Beschriftung, Farbe, ...) durch Festlegen von Eigenschaften (Properties) gestaltet wird und die Ereignismethoden als Reaktion auf diese Ereignisse erstellt werden.	Programmierkonzept benutzen, erste Lösungsstrategie entwerfen
Der Unterricht beschränkt sich bei den Einstiegsaufgaben auf wenige, typische „visuelle“ Komponenten (z. B. Fenster, Schaltfläche, Bildfeld) und Ereignisse (z. B. Maus-, Tastatur-Ereignis). Zentrale Objekte sind Fenster, die als Klassen in Modulen realisiert werden. Die *Grundelemente der klassischen Algorithmik* werden im Zusammenhang mit der Funktionalität der Ereignismethoden dieser Fensterklassen entwickelt. Entscheidungen erhalten bei der Auswertung der eintreffenden Ereignisse dabei eine besondere Bedeutung.	Reduziertes Modell definieren, problembezogene Objekte spezifizieren
Durch *Vererbung* (Ableitung aus einer Oberklasse) wird eine neue Fensterklasse erzeugt, die sich von der vorgebenen Oberklasse durch die verwendeten Komponenten, Ereignisse und Methoden unterscheidet.	Lösungskonzept implementieren
Die Problemanalyse ergibt eine Modellierung der jeweiligen Problemkreise in diesen Kategorien. Auf dieser Grundlage wird ein Ablaufschema für die einzelnen Arbeitsschritte entwickelt. Die Beschreibung und Dokumentation der Funktionalität der verwendeten Klassen (z. B. in Diagrammen) bewegen sich in diesem Rahmen.	Denkschema entwickeln, Lösungen dokumentieren

Graphik-Konzept und selbst definierte Methoden (20-25 Std.) Projekte mit graphischen Aufgabenstellungen eignen sich für den Anfangsunterricht, weil mit der Ausgabe eines „graphischen Ergebnisses“ wesentliche Teile der Programmverifizierung durchgeführt werden.	
Aus der Sicht der Anwenderin bzw. des Anwenders visueller Werkzeuge werden die Systemumgebung selbst sowie die verfügbaren kommerziellen Produkte immer eine Vorlage für die eigenen Konzeptionen sein. Die *Analyse und Bewertung* dieser Produkte, z. B. von Zeichenprogrammen, zeigt zum einen die charakteristischen Merkmale, aber auch die Möglichkeiten und Grenzen der Werkzeuge auf. Diese Merkmale können dann, in reduziertem Umfang, Grundlage der eigenen Projekte sein. Erweiterungen der reduzierten Funktionalität des eigenen Projekts schaffen Raum für Binnendifferenzierung.	Anwendungssoftware analysieren, Software bewerten, Problemlösungen weiterentwickeln
In ereignisorientierten Systemen ist die Bedeutung der Kontrollstruktur „Wiederholung“ stark reduziert. Die Problematisierung der Ereignisschleife des Systems bleibt einem vertiefenden Zugang vorbehalten.	Standardlösungen anwenden
Die Komponenten-Klassen besitzen Attribute (interne Zustandsvariable) und Dienste (Methoden), die ihre Funktionalität beschreiben. Das Prinzip der *Datenkapselung* verbietet einen direkten Zugriff. Durch Methoden und Eigenschaften (Properties) ist ein kontrollierter lesender oder schreibender Zugriff möglich (Anfragen bzw. Aufträge).	Programmierkonzepte verstehen, Strukturen analysieren
Erweiterte Aufgabenstellungen sowie die Grundprinzipien der *Modularisierung* erfordern die Verwendung eigener, selbst definierter Variablen sowie selbst definierte Methoden. Die Prinzipien der OOP erfordern eine Beschränkung auf die Verwendung von Variablen als Zustandsvariablen oder lokale Hilfsvariablen.	Programmierkonzepte verstehen
Grundprinzipien der OOP (15-20 Std.) - *Objekte reagieren autonom auf Aufträge oder Anfragen (Datenkapselung)* - *Variablen und Methoden sind an die Objekte gebunden.* - *Objekte stehen in Beziehungen zueinander, die analysiert und modelliert werden.*	Sprachrachkonzepte charakterisieren
Die verwendeten Entwicklungswerkzeuge enthalten als Hybrid-Werkzeuge oft einem imperativen Kern, der um objektorientierte Bestandteile erweitert wurde. Daher kann in solchen Werkzeugen gegen die Grundprinzipien der OOP verstoßen werden. Die Reflektion über diese Alternativen und deren Bewertung ergibt eine Beschreibung der charakteristischen Merkmale dieser Sprachkonzepte, die in der folgenden Jahrgangsstufe den konstruktiven Überlegungen zugrunde liegen.	Sprachkonzepte beurteilen

Numerische und alphanumerische Daten	
Die Wertebereiche der in den Klassen vorgegebenen Eigenschaften und der Parameterlisten der Methoden vermitteln von Beginn an Kenntnisse über elementare Daten und weitere in der Klassenhierarchie vorgegebene nicht-visuelle Klassen. Die *systematische Behandlung numerischer und alphanumerischer Daten* knüpft daher an die bisherigen Erfahrungen der Schülerinnen und Schüler an. Ein- und Ausgaben werden erweitert auf numerische und alphanumerische Werte. Die erforderlichen Konvertierungen werden eingeführt und reflektiert. Bei der Konstruktion von Methoden tritt ihr Ein-/Ausgabeverhalten in den Vordergrund.	Standardlösungen kennen lernen und anwenden, Lösungen bewerten
Die durchgehend im Unterricht verwendeten Sprechweisen der OOP sind besonders geeignet, die einzelnen Bestandteile der graphischen Benutzerschnittstelle (GUI) zu beschreiben. Es ergibt sich daher in natürlicher Weise eine objektorientierte Beschreibung des Rechnersystems. Der Computer stellt sich zudem auf dieser Ebene als ”reagierender Automat” dar. Diese Sichtweisen können in einem späteren, tieferen Verständnis als *von-Neumann-Rechner* bzw. endlicher Automat konkretisiert werden.	Rechnersysteme in objektorientierter Terminologie charakterisieren
Die einzelnen Aufgabenstellungen sind zunehmend *projektorientiert* und werden inkrementell entwickelt, dokumentiert und bewertet.	Problemlösungen optimieren

(4) Sequenz „wissensbasiert“

Sequenz für die Jahrgangsstufe 11 mit wissensbasiertem Ansatz

Lernsequenz	Verknüpfung mit der Obligatorik
Einführung in die Arbeitsweise wissensbasierter Systeme (9-12 Std.)	
Ausgehend von aktuellen Problemen (z. B. Schachcomputer Deep Blue, automatische Hotelbuchungssysteme u.a.) zur maschinellen Intelligenz werden *Grundzüge von maschineller Wissensverarbeitung* untersucht und die Begriffe Wissen, Information und Daten abgeklärt. Zu einfachen Fragestellungen wird dann das Wissen über gewisse Objekte in Form von Fakten abstrahiert und schrittweise von umgangssprachlichen Formulierungen in die logische Programmiersprache PROLOG übertragen. Die Arbeitsweise und Bedienung der PROLOG-Umgebung wird erläutert und es werden durch die Maschine erste Schlussfolgerungen vollzogen. *Deklarative* und *prozedurale* Sichtweisen werden unterschieden. *Prädikate* und deren Argumente, *Anfragen*, PROLOG-Variablen und die *Unifikation* werden eingeführt. *Funktoren* können zur Strukturierung von Objekten eingesetzt und erste Datenkopplungen über mehrere Prädikate ausgeführt werden.	Anwendungssituationen zur Motivation und Veranschaulichung, Analyse von Anwendungsbereichen, Entwicklung von Datenorganisationen, Überprüfung von Lösungskonzepten, Analyse und Vergleich von Funktionsweisen, überwiegend gelenktes Unterrichtsgespräch und praktisches Arbeiten am Rechner

Modellierung von Wissensbereichen mittels Fakten und Regeln (25-30 Std.) Die *Syntax* und *Semantik* der Sprache PROLOG wird genauer geklärt und zur Modellierung von umfangreicheren Wissensbereichen herangezogen. Dazu gehören Regeldefinitionen, die Stelligkeit von Prädikaten, die Und- und Oder-Verknüpfungen, der Gebrauch von Variablen und anonymen Variablen in Regeln und Anfragen. Zur Verfolgung der Arbeitsweise des PROLOG-Beweisers und zur besseren Fehlersuche wird in Modelle zur Ablaufverfolgung und Veranschaulichung eingeführt. Datenverknüpfungen über mehrere Datenbasen erlauben die Konstruktion von „neuem“ Wissen und können auch zur Aufhebung des Datenschutzes (Deanonymisierung) z. B. bei Befragungen eingesetzt werden. Spezielle Prädikate erlauben das Hinzufügen und Löschen von Klauseln und ermöglichen damit die nicht unproblematische dynamische Veränderung von Wissensbasen. Ein kleines *Expertensystem* kann aufgebaut werden. Bei der Einführung von Prädikaten zur Ein- und Ausgabe über Tastatur und Bildschirm ist ein *Vergleich mit imperativen Sichtweisen* sinnvoll. In Abhängigkeit von den gewählten Problemen wird man auch auf Operatoren, Terme, Werte und das Rechnen mit PROLOG und deren logischer Interpretation eingehen.	Kennen lernen formaler Sprachen und Grammatiken, Verstehen von Programmierkonzepten, Spezifizieren von problembezogenen Objekten und ihrer Wechselwirkungen, Entwickeln und Überprüfen von Lösungskonzepten, Erkennen von Chancen und Risiken, Entwicklung von Verantwortungsbewusstsein im Umgang mit Daten, Vergleich von Konzepten, selbständiges Arbeiten am System, erste Gruppenarbeiten
Rekursive Prädikate, Listenstrukturen und Suchverfahren (20-25 Std.) *Rekursive Prädikatdefinitionen* werden hier über die Vorfahrdefinition in einem Familienstammbaum eingeführt und unmittelbar mit der *Datenstruktur Liste* vertieft. Die Einführung der Liste orientiert sich an der generellen Konzeption von Datenstrukturen in PROLOG, die diese immer auf *Baumstrukturen* zurückführen. Grundprädikate zur Listenverarbeitung werden herausgearbeitet und im Zusammenhang mit kleineren Problemen eingeübt. Zur Vertiefung bieten sich systematische Suchverfahren zur Pfadsuche in einem Graphen an, die im Zusammenhang mit kleinen Strategiespielen, Planungsaufgaben oder der Mustererkennung auf die grundlegenden Verfahren der Tiefen-, Breiten- und heuristischen Suche führen.	Kennen lernen und Anwenden allgemeiner Strategien und Standardverfahren, Entwickeln leistungsfähiger Lösungskonzepte, Analyse von Anwendungssituationen, Einsatz leistungsfähiger Lösungskonzepte, vermehrte Gruppenarbeit
Einblick in die maschinelle Sprachverarbeitung (16-20 Std.) Als Einstieg bieten sich hier Zeitungsartikel über Erfolge oder Misserfolge der maschinellen Übersetzung der geschriebenen und gesprochenen Sprache an. Nach einer Einführung der linguistischen Begriffe Syntax, Semantik und Pragmatik werden elementare Techniken der *syntaktischen Sprachverarbeitung* erarbeitet und auf kleinere formale Sprachen angesetzt. Eine Beschreibung in Form von Syntaxdiagrammen kann dann unmittelbar in ein kleines Projekt für einen Interpreter oder Parser der jeweiligen Sprache münden.	Anwendungen und deren Grenzen erkennen, Beurteilen von Syntaxregeln und Beschreibungssystemen, Lösungsstrategien zunehmend selbständig entwickeln, fachübergreifende Sichtweisen, erste Projektarbeitsformen

(5) Sequenz „funktional“

Sequenz für die Jahrgangsstufe 11 mit funktionalem Ansatz

Der funktionale Ansatz geht zurück auf eine der ältesten Programmiersprachen überhaupt: LISP. Ihr Einsatzgebiet ist hauptsächlich das der künstlichen Intelligenz. Von den vielen heute zu findenden funktionalen Sprachen (SASL, KRC, MIRANDA, ...) hat vor allem der LISP-Dialekt SCHEME im pädagogischen Raum größere Bedeutung erlangt. Trotz der Nähe von LOGO zu LISP scheint eine Verwendung dieser Sprache aufgrund ihrer Beschränkungen nicht sinnvoll.

Lernsequenz	Verknüpfung mit der Obligatorik
Die Technik des Abstrahierens mit Funktionen (20-25 Std.)	
Nutzt man eine der attraktivsten Eigenschaften von LISP, dass es im Gegensatz zu den meisten anderen Programmiersprachen interaktiv ist, so können die Schülerinnen und Schüler von Anfang an eigenständig experimentell die funktionale Denkweise kennen lernen. Sie kombinieren *atomare* (arithmetische) *Ausdrücke* zu beliebig komplexen *Baumstrukturen* und erweitern das System um selbst definierte Funktionen, wobei sie das *Prinzip der Bindung von Ausdrücken an Namen* ebenso kennen lernen wie das Auswerten von Präfixausdrücken (*Auswertung durch Ersetzen*). Parallel dazu wird der *Begriff der Funktion* geklärt und als zentral herausgestellt. Ein Alltagsphänomen etwa aus der Regelungstechnik (Heizungsanlage) wird unter dem funktionalen Aspekt analysiert. Als Problemlösestrategie wird erkannt, dass Funktionen durch Kombination aus gegebenen Grundfunktionen zu abstrahieren sind, die das gewünschte Verhalten modellieren. Der Begriff der *Umgebung* als Menge aller zur Problemlösung nötigen Funktionen wird im direkten Umgang intuitiv klar und dann exakt definiert.	Daten und Algorithmen abstrahieren Abschnitt 1: Speziell Technik der Kombination und Abstraktion; Aufbau von Umgebungen
Dies kann an ersten größeren Beispielen nach der *Top-Down-Methode* geübt werden. So kann ein Paket zur Berechnung der Oberflächen und Volumina verschiedener Körper erstellt werden, wobei diese als aus einfachen Figuren (Dreieck, Rechteck, ...) zusammengesetzt erscheinen. Die Hierarchie der erstellten Funktionen entspricht einer möglichen Hierarchie der geometrischen Objekte. Dabei wird besonders genutzt, dass anders als etwa in PASCAL die Definition einer Funktion auch solche Funktionen enthalten darf, die noch gar nicht definiert sind. So lässt sich die Arbeit vom allgemeinen zum speziellen durchführen, bis man bei bereits vorhandenen Funktionen bzw. Atomen angelangt ist. Zur *Ablaufsteuerung* (Fehleingaben (negative Längen, ...)) werden Konstrukte zur *Fallunterscheidung* verwendet.	Lösungskonzepte in einer Programmiersprache entwickeln: Auswerten von Ausdrücken in Umgebungen; Anwenden von Funktionen auf Argumente

Abstraktion mit Daten - Listen und hierarchische Strukturen (30-35 Std.) Das Problem der Zusammenfassung zweier Zahlen (*Paarbildung)* zu einem neuen Datum tritt auf, wenn man ein Arithmetikpaket für die Bruchrechnung zur Verfügung stellen will. Das Rechenpaket selber wird nach der Top-Down-Methode erstellt. Dabei wird darauf geachtet, dass die vom System bereitgestellten *Konstruktoren* und *Selektoren* für Paare nur auf der untersten Ebene benutzt werden. So entsteht eine Trennung zwischen Anwenderebene (Funktionen, die die Anwenderin bzw. der Anwender zur Nutzung bekommt) und einer Entwicklerebene (Funktionen, die die elementaren Konstruktoren und Selektoren enthalten). Das Paket kann durch das Hinzunehmen anderen Zahltypen erweitert werden. Das Prinzip genügt, um Daten aus den verschiedensten Bereichen in einer *Liste* zusammenfassen zu können. Organisationspläne von Firmen, Stammbäume, Klassifikationen (etwa in der Biologie) usw. verwenden *hierarchische Strukturen* auf der Grundlage von Listen. Ein solches Beispiel wird analysiert und eine Umgebung implementiert, in der die Struktur dargestellt werden kann. Die Liste wird als universell verwendbare *Datenstruktur* herausgestellt. Auf der Grundlage der o.a. Problemlösestrategie sind Grundoperationen (Elementanzahl in einer Liste, n-tes Listenelement, Umdrehen von Listen, „Glätten“ von Listen, ...) auf allgemeinen Listen zu definieren. Diese arbeiten rekursiv. Gerade *rekursive Funktionsdefinitionen* entsprechen häufig den umgangssprachlichen Formulierungen. Dabei *sind lineare Rekursion* und *Baumrekursion* zu unterscheiden. Die jeweiligen Beispiele arbeiten zunächst auf glatten Listen, werden dann schrittweise auf beliebige Strukturen ausgeweitet. Die *Standardkonstruktoren* für Listen werden behandelt, *Vergleichsoperatoren* für Listen und Atome eingeführt. Nun kann der Begriff des *abstrakten Datentyps* eingeführt werden. Der Typ Menge etwa mit den nötigen Operationen wie Durchschnittsbildung, ... wird als (unsortierte) Liste implementiert	Daten und Algorithmen abstrahieren Abschnitt 2: Kopf-Restliste-Methode; Rekursion
Suchen und Sortieren in hierarchischen Strukturen - Ergebnislisten im Sport (25-30 Std.) Das Problem der Erstellung von Ergebnislisten im Sport (oder etwa ein Vokabellernprogramm) benötigt zur Realisierung nicht nur eine systematische Anwendung der erlernten Techniken: Vielmehr müssen neue Ergebnisse in die schon bestehende Liste einsortiert werden, diese muss ggf. nach verschiedenen Kriterien ausgegeben und durchsucht werden. Dabei werden verschiedene Algorithmen für *Such- und Sortierprobleme* auf zunehmend komplexer werdenden Strukturen implementiert und bzgl. des *Effizienzverhaltens* verglichen. Die Funktionen werden wieder rekursiv auf Listen arbeiten. Eine Ausweitung dieser Probleme auf allgemeinere Datenstrukturen wie *Bäume* kann erfolgen, indem man einen Sportler in verschiedenen Disziplinen antreten lässt.	Daten und Algorithmen abstrahieren Abschnitt 2: Allgemeine Datenstrukturen; Suchen; Sortieren
Sprachen, Grammatiken, Technik in unserer Gesellschaft (15-20 Std.) Ausgehend von dem direkten Umgang mit einem Eliza-Programm wird der *Einfluss der Technik auf die Gesellschaft und den Einzelnen* unter Zuhilfenahme von Texten (Zeitschriften, Internet) beleuchtet. Zum anderen kommt aber auch das einem solchen Programm zugrunde liegende informatische Problem als Problem der *Manipulation von Wortlisten* zur Sprache. Ein gutes Anwendungsbeispiel für solche Manipulationstechniken, die auf der Anwendung von *Grammatiken* beruhen, sind Lindenmayer-Systeme, die zur Beschreibung fraktaler Systeme entwickelt wurden (Graftale). Funktionen sowohl *zur Substitution in Listen* als auch deren graphische Darstellung werden implementiert.	Typische Einsatzbereiche Algorithmen, Sprachkonzepte: Beschäftigung mit Grammatiken, formalen Sprachen

(6) Sequenz „anwendungsorientiert“

Sequenz für die Jahrgangsstufe 11 mit anwendungsorientiertem Ansatz unter Verwendung von Datenbankwerkzeugen

Lernsequenz	Verknüpfung mit der Obligatorik
Datenbankbeispiele (3-4 Std.)	
Die Schülerinnen und Schüler lernen *Datenbanken* aus dem Alltagsbereich wie z. B. Telefondankenbanken, Versandhauskataloge auf CD oder Datenbanken aus dem Bereich der Lexika kennen. Es werden der häufig riesige Informationsgehalt sowie die vielfältigen Such- und Recherchemöglichkeiten vorgestellt. In diesen Zusammenhang fällt auch die Nutzung des Internet.	Typische Einsatzbereiche, Möglichkeiten, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen
Bei der Vorstellung z. B. einer Telefon-CD werden erste Fragen zum Thema *Schutz der Daten* vor Missbrauch aufgeworfen. Kernaussagen des Datenschutzgesetzes können thematisiert und vorgestellt werden. Gleichzeitig werden erste Fragen zum Thema Schutz personenbezogener Daten vor Missbrauch (kurz, aber missverständlich *Datenschutz* genannt) aufgeworfen.	Anwendung als Ausgangspunkt zur Motivation die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen
Außerdem wird der Umgang mit dem PC eingeübt.	
Abfragen und Ansichten in Datenbanken (15-18 Std.)	
Die Formular und Tabellenansicht werden vorgestellt und für verschieden Übungen genutzt.	Problemstellungen eingrenzen und Probleme strukturieren
Es wird eine Beispieldatenbank z. B. der Datenbestand einer Bibliothek verwendet, welche bereits gefüllt vorliegt. Dazu wird ein ausreichend komfortables Datenbankmanagementsystem verwendet, welches alle benötigten Funktionen enthält.	
Datensätze sind einzufügen und zu löschen. Um mögliche Ergebnisse besser überblicken zu können, werden die *Projektion* und die *Selektion* eingeführt. Dabei sind gleichzeitig sowohl Abfragen mit QbE sinnvoll als auch eine Einführung in SQL, um u.a. die *Abfragetechniken* zu verallgemeinern und von einem speziellen Datenbanksystem abzuheben. Außerdem lassen sich so erste *algorithmische Aspekte* deutlich machen. Es werden sowohl Auswahlabfragen als auch Funktions-, Update- und Löschabfragen behandelt. An dieser Stelle wird der Unterschied zwischen Zuweisung und Vergleich in Syntax und Semantik deutlich.	Entwickeln und Überprüfen von Lösungskonzepten, ein Lösungskonzept als Denkschema entwickeln, allgemeine Strategien und Standardlösungen kennen lernen und verstehen
Datenbank-Entwurf (25-28 Std.)	
Mit der Erstellung neuer Tabellen in bereits bestehenden Datenbanken werden erste *Aspekte der Modellierung* angesprochen. Dazu gehören Gesichtspunkte der *funktionalen Abhängigkeiten* der Datenfelder, der sinnvollen *Festlegung der Schlüsseleigenschaften* und der *Indizierung* sowie *Überlegungen zum Datentyp der definierten Felder*. Bei der Untersuchung der funktionalen Abhängigkeiten ist es sinnvoll, die *Normalformen* einzuführen, um so u.a. zur Erfüllung von Integritätsbedingungen zu kommen und Update-Fehler zu vermeiden. Mit Hilfe des Join als Verbindung verschiedener Tabellen wird das *Prinzip relationaler Datentypen* deutlich. Tabellen stehen sowohl für Entitys als auch für Relationen.	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren, Probleme eingrenzen und Probleme strukturieren, Anforderungen an ein Modell aufstellen, ein Lösungskonzept als Denkschema entwickeln, Lösungen nach vorgegebenen Kriterien bewerten, den Einsatz von Informationssystemen kennen lernen und verstehen
Beim Entwurf einer Datenbank wird festgelegt, welche Anforderungen der Nutzer an das System stellt. Dabei wird deutlich, dass die Anwenderin bzw. der Anwender in der Regel nicht die vollständige Sicht auf die Datenbank hat. An dieser Stelle wird auf den Begriff der *Drei-Ebenen-Architektur* eingegangen.	Den Einsatz von Informations- und Kommunikationssystemen in verschiedenen Bereichen untersuchen und bewerten, den Strukturwandel in Industrie und Gesellschaft erkennen und beschreiben

Sinnvollerweise lässt sich an dieser Stelle eine praktische Erkundung durchführen, z. B. in einer öffentlichen Bibliothek, um den professionellen Datenbankbetrieb zu studieren und anknüpfend an das Arbeiten im unterrichtlichen Datenbanksystem mit den vielfältigen Bearbeitungsmöglichkeiten auch die eingeschränkte Sicht der Benutzer im professionellen Datenbankbetrieb kennen zu lernen.	Allgemeine Strategien und Standardlösungen kennen lernen: Systematiken und Theorien zur Lösung spezifischer Anwendungssituationen Problemstellungen eingrenzen und Probleme strukturieren, ein Informatikmodell gewinnen: Probleme eingrenzen und spezifizieren, reduzierte Systeme definieren
Zusätzlich wird das relationale Datenbankmodell mit dem *Netzwerk*- und dem *hierarchischen Modell* verglichen.	Lösungskonzepte implementieren und testen, Lösungen dokumentieren, Modularisierung
Der Abschnitt wird mit der Modellierung einer vollständigen Datenbank abgeschlossen, sodass alle Phasen des Entwurfs durchlaufen werden.	Selbstständige und projektorientierte Arbeitsformen
Anwenderprogrammierung (24-30 Std.) Die verschiedenen Sichtweisen einer Datenbank erfordern den Einstieg in die *Datenbank-Programmierung*. Dazu werden alle Möglichkeiten einer modernen Programmiersprache genutzt. Dabei werden Variablen benutzt sowie die Verarbeitungsaufgaben strukturierend und modular mit Hilfe von Prozeduren gelöst. Unerlaubte Dateneinsicht und Datenänderung im Zusammenhang mit dem Datenschutz führt zur Behandlung von Möglichkeiten der Zugriffs- und Identitätskontrolle. Dieser Abschnitt kann nur *exemplarisch* in die Datenbankprogrammierung einführen, eine Vertiefung erfolgt in Jahrgang 12 und 13.	Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln; Daten und Algorithmen abstrahieren: allgemeine Strategien und Standardlösungen kennen lernen und anwenden, Programmierkonzepte allgemeiner und spezieller Art verstehen und benutzen, Formen des Strukturierens einsetzen, ein Lösungskonzept als Denkschema entwickeln, Lösungen dokumentieren
Mehrbenutzerbetrieb (17-20 Std.) Es werden Probleme des *Mehrbenutzerbetriebs* angesprochen. Hierzu gehören die inkonsistente Sicht und die inkonsistente Datenbank sowie die Folgen von Systemabstürzen. Es werden der *Transaktionsbegriff* und der *Transaktionsmanager* eingeführt und es wird erarbeitet, wie Untersuchungen zur Serialisierbarkeit von Datenbankzugriffen die Probleme lösen können. Verschiedene *Synchronisationsverfahren* wie z. B. das Sperrverfahren und das optimistische Verfahren werden behandelt. Auch das Problem des Systemzusammenbruchs wird angesprochen, dabei werden Lösungsmethoden als Recovery-Strategien vorgestellt.	Aufdecken der Funktionsweise bekannter Werkzeuge, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen, Kommunikations und Vernetzungsstrukturen einordnen/Entwickeln leistungsfähiger Lösungskonzepte

3.4.2 Sequenzen für die Jahrgangsstufen 12 und 13

Nachdem wesentliche Gebiete des Schulfachs Informatik in der Klasse 11 einführend erschlossen und bis zu einer ersten Stufe qualitativer Arrondierung aufgearbeitet worden sind, wird sich die Qualifikationsphase einerseits der Vertiefung und Verbreiterung der fachinhaltlichen und fachmethodischen Basiskenntnisse und andererseits der Erschließung neuer Felder der informatischen Theorie, ihrer unterrichtlichen Aufarbeitung und produktorientierten Nutzbarmachung im Rahmen projektorientierten Lernens zuwenden müssen.

Unterrichtlich bewährt hat sich eine spiralförmige Vertiefung der charakteristischen Ansätze des in Klasse 11 favorisierten Paradigmas unter dem Leitaspekt der „Sprach-“ bzw. „Konzepterweiterung“ über die jeweils durch die Programmiersprache vorgegebene Plattform hinaus.

Zusätzlich muss dann aber wenigstens ein andersartiges neues Schwerpunktthema aufgegriffen werden, das nicht mehr als Fortsetzung der paradigmatischen Differenzierung aus der Startphase angelegt ist. Für die Schülerinnen und Schüler wird der Blick auf gänzlich neue Sachgebiete der Informatik gerichtet und für die Unterrichtenden wird die Auswahl erweitert, um die Flächen des „didaktischen Würfels“ unter Beachtung der fachlichen Obligatorik inhaltlich und methodisch variantenreich füllen zu können.

Gegen Ende der Qualifikationsphase wird der Unterricht auf Themen ausgerichtet, die eine Zusammenführung der Kenntnisse aus den unterschiedlichen Schwerpunktthemen zum vorrangigen Ziel hat und damit bei der Schülerin bzw. beim Schüler kumuliertes und vernetztes Wissen und Können anlegt.

Ein gleichmäßig wachsendes Anspruchs- und Komplexitätsniveau führt zu breiter angelegten Fragestellungen, praxisrelevanteren Unterrichtsergebnissen und zu einer zunehmend soliden Reflexionsmöglichkeit informatischen Arbeitens. In dieser Weise entwickelt sich der Unterricht der Qualifikationsphase stückweise auf wesentliche Teilqualifikationen zu, die schließlich in der Abiturprüfung nachzuweisen sind.

Die folgende Tabelle liefert eine wenn auch noch grobe Orientierung für die Festlegung auf eine spezielle Lernsequenz in der Qualifikationsphase:

Zeit	Phase	Schwerpunktthema	Aspekte im Unterricht
11/I – 11/II	Grundsätzliche Orientierung und Vermittlung von informatischen Arbeitstechniken	Algorithmische Grundschulung und Erwerb informatischer Basiskenntnisse im Rahmen eines ausgewählten Paradigmas	Lösung informatischer „Grundaufgaben“ durch angeleitetes Analysieren, Modellieren, Konstruieren und Bewerten
12/I – 13/I	Spiralförmige Vertiefung (ggf. noch innerhalb des gewählten Paradigmas)	Konzept- und Spracherweiterung durch die (Fort)Entwicklung von Modellen und die Schaffung eigener neuer Sprachmittel jenseits programmiersprachlicher Vorgaben	Z.B.: ADT-Konzept im imperativen Ansatz, Entwicklung neuer Klassenhierarchien im objektorientierten Paradigma
	Erschließung neuer Themenfelder	Maschinennahe Konzepte	Intramaschinelle Organisation und Funktion, maschinennahe Sprachen, Sprachübersetzer, Betriebssysteme
		Netzstrukturen	Intermaschinelle Kommunikation in lokalen und globalen Netzen, Chancen und Risiken im Internet
		Ausgewählte Gebiete der theoretischen Informatik	Automatentheorie, Schaltnetze und -werke, Turingmaschine und Berechenbarkeit, Theorie formaler Sprachen
13/II	Vorbereitung auf die Abiturprüfung	Vernetzung von Themenfeldern	Übergeordnete Aspekte und zusammenfassende Generalisierungen, die schlusssteinartig verschiedene Schwerpunktthemen in sich aufnehmen
		Reflexion informatischen Arbeitens	Assessment von Methoden und Produkten im Rückblick auf voraufgegangene Projekte
		Formen fachlichen Diskurses	Ausgearbeiteter Vortrag (z. B. über Facharbeit), sog. mündliche Prüfung

3.4.2.1 Vertiefung eines paradigmatischen Ansatzes

Im Folgenden soll eine Verfeinerung der Schwerpunktthemen, eine Anbindung an die Obligatorik und eine Verknüpfung mit den Kategorien der drei Flächen des „didaktischen Würfels“ hergestellt werden.

Da die spiralförmige Vertiefung (ggf. zu Beginn der Qualifikationsphase) i.A. noch innerhalb des einmal gewählten Paradigmas oder - in bewusster Abgrenzung - vor dem Hintergrund eines alternativen Sprachansatzes unterrichtet wird, bietet sich für diesen Abschnitt noch eine Fortsetzung der paradigmatisch differenzierten unterrichtlichen Sequenzierung an.

Die „Erschließung neuer Themenfelder“ ist hingegen nicht mehr an eines der vorgestellten Paradigmen gebunden und stellt damit ein thematisches Angebot dar, auf das alle bisherigen unterrichtlichen Planungen gleichermaßen zurückgreifen können. Bei der konkreten Entschei-

dung für oder gegen einzelne der (hier vorgeschlagenen) Schwerpunkthemen können die Unterrichtenden den Freiraum zur Weiterentwicklung ihrer individuellen Themenpalette nutzen.

Da die Obligatorik eine rigide Festlegung auf einzelne Themenaspekte bewusst vermieden und stattdessen eine Ausrichtung am „didaktischen Würfel“ bevorzugt hat, ergeben sich Lernsequenzen nicht zwangsläufig deduktiv, sondern eher als Vorschlag aus möglichen Alternativen. Statt die Erkundung des „informatischen Geländes“ in der Qualifikationsphase auf eine spezielle Streckenführung einzuengen, wurden durch die Festlegung einer allgemeineren Obligatorik markante Wegepunkte genannt, an denen man sich jeweils wieder zu treffen hat. Die Festlegung der speziellen Route bleibt dabei für jeden Kurs eine eigenständige Aufgabe im Rahmen der didaktischen Entscheidungen der Unterrichtenden.

Unabhängig von der einzelnen Entscheidung haben die Kurse der Jahrgangsstufe 11 im Rahmen der Bearbeitung einer Reihe von Anwendungsfeldern der Informatik zu einer algorithmischen Grundschulung im Rahmen des ausgewählten Paradigmas geführt. Die abstrakten Modellierungs- und die sprachabhängigen Codierungsfähigkeiten sind so weit entwickelt, dass die im Rahmen des jeweiligen Sprachansatzes verfügbaren Werkzeuge zielgerichtet und erfolgreich zur Problemlösung herangezogen werden können. Im Sinne eines spiralförmig angelegten Lernprozesses soll dies nun vertieft und erweitert werden, indem vorrangig Problemstellungen in den Blick genommen werden, deren Lösungen durch die verfügbaren Werkzeuge nicht mehr ohne weiteres zugänglich sind.

Detaillierter entwickelt wird dies im Folgenden an verschiedenartigen Unterrichtssequenzen:

(1) Sequenz „imperativ“

Vertiefung des imperativen Ansatzes

Lernsequenz	Verknüpfung mit der Obligatorik
Für den klassisch-imperativen Weg wird man Techniken der komplexen Datenorganisation zu vermitteln haben. Die *Konzeption und Implementation Allgemeiner Datentypen* (*ADT*s) als Erweiterung des verfügbaren Sprachrahmens wird zum übergeordneten Thema, wenn man dynamische Strukturen wie *Schlangen*, *Stapel* und *allgemeine Listen* erarbeitet. Zirkuläre Strukturen wie *Ringe* können den Übergang bilden zu den klassisch *mehrfach verzweigten Organisationsformen*: Der *binäre Baum* - als Ahnenbaum ohne Vergleichskriterium oder als *Suchbaum* unter Einbeziehung einer Ordnungsrelation - gehört zum Standarddurchgang des Unterrichts, wobei Verfahren effizienter *Auslagerung* solcher Strukturen *auf externe Speichermedien* nicht außer Acht gelassen werden sollten. Eine vertiefte Auseinandersetzung mit allgemeineren und damit auch *komplexeren Baumstrukturen* bis hin zu *allgemeinen Graphen* kann nach Belieben auch ans Ende der Qualifikationsphase und damit in die letzte Spiralwindung des Lehrplans platziert werden. Bei der Erarbeitung der o.g. dynamischen Strukturen wird man auf Fragen der *Neukonzeption durch Spezialisierung oder Redefinition* und damit auf eine *Brücke* stoßen, die *ins Feld des objektorientierten Paradigmas* führen kann. Ob man sich dann auf wesentliche Facetten dieses Ansatzes beschränkt oder gänzlich auf diese Modellierungs- und Implementierungstechnik umsteigt, bleibt wiederum eine Entscheidung für den speziellen Unterrichtsgang.	Modellieren und Konstruieren, dabei insbesondere: Daten und Algorithmen abstrahieren und Lösungskonzepte weiterentwickeln Lernen im Kontext: Zugriff auf große Personenregister per Suchbaum, Computersimulation eines Rangierbahnhofs durch Stapel und Schlangen Fachspezifische Vorgehensweise: Entwurf einer Datenorganisation, Entwicklung neuer Werkzeuge

(2) Sequenz „objektorientiert allgemein“

Vertiefung des objektorientierten Ansatzes unter Verwendung plattformunabhängiger Klassenbibliotheken

Lernsequenz	Verknüpfung mit der Obligatorik
Taschenrechner	
1. OOA Ein Taschenrechner soll auf dem Bildschirm simuliert werden. Zunächst werden die zugehörigen Objekte und Klassen entdeckt. Dazu werden vorher mögliche Arten von Objekten vorgestellt. Zum Entdecken der Klassen eignen sich CRC-Karten.	Objektorientierte Analyse Ein informatisches Modell gewinnen Klassifizierung: Problembezogene Objekte und ihre Wechselwirkungen spezifizieren, Nutzen fortschreitender Analysetechniken, selbständige Arbeit in Gruppen
2. OOD Vererbungsstrukturen und das Verwenden bekannter Klassen werden berücksichtigt. Es ergeben sich eine Klasse für die Anzeige, eine für den eigentlichen Rechner und eine Vererbungshierarchie für die Tasten mit Eingabe-, Funktions- und Operatortasten. Eine genaue Dokumentation, auch unter Verwendung von Diagrammen (z. B. Coad-Yourdon) ist wichtig. Weil alle Tasten auf Mausdruckereignisse reagieren müssen, wird das Konzept einer Liste aller Objekte als *„Ereignis-Bearbeiter“* und einer *„Rundruf-Anwendung“* entwickelt. Dadurch werden die Konzepte *„lineare Liste“* und *„Ereignisanwendung“* vertieft.	Objektorientiertes Design Ein informatisches Modell gewinnen Verwenden von vorhandenen Klassenbibliotheken, Objekt- und Klassenbeziehungen Objektdiagramme Entwicklung einer Datenorganisation
3. OOP Eine erste Version des Taschenrechners enthält nur die Anzeige und funktionsfähige Eingabetasten. Die Klasse Rechner schickt nur die Ergebnisse der Eingabetasten an die Anzeige.	Objektorientiertes Programmieren Ereignisgesteuerte Programmierung, Lösungskonzepte implementieren und testen, Gruppenarbeit
4. Ein UPN-Rechner Ein funktionsfähiger Rechner wird erstellt, der nach der UPN-Methode arbeitet. Dazu wird die Klasse *Stapel* von der Klasse Liste abgeleitet.	Umgekehrte Polnische Notation Problembezogene Objekte und ihre Wechselwirkungen spezifizieren, Problemlösungen weiterentwickeln
5. Ein Rechner mit algebraischer Eingabelogik In einer weiteren Stufe schreiben alle Tasten in die Anzeige. Ein Druck auf die “=”-Taste löst die Auswertung des angezeigten Terms aus. Auf Funktionstasten sollte verzichtet werden. Dazu wird ein *Termbaum* benötigt; die Datenstruktur *Baum* und die nötigen *Durchlaufalgorithmen* werden entwickelt.	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren

Dateiverwaltungsprojekt	Effektive Organisation größerer Datenansammlungen (Datenbanken)
Bestimmte Daten sollen verwaltet werden. Welche Daten verwaltet werden, sollte von der konkreten Unterrichtssituation abhängen. Gut ist es, wenn an reale Probleme wie z. B. Schülerbüchereiverwaltung angeknüpft werden kann. Alternativ können die „Surfdaten"-Aufzeichnungen eines Kommunikationsservers herangezogen werden (Wer war wann wo und wohin „ging" er anschließend), Scannen von Mails auf Schlagwörter (Absprache zum Ladendiebstahl? zum Schummeln? ...) und Aufbau einer Volltextdatenbank. Praktische Beispiele aus der Wirtschaft (Betriebsdatenerfassungssysteme) ermöglichen besonders gut die Diskussion von Fragestellungen des Datenschutzes.	Lernen im Kontext: Reale Dateiverwaltungsanwendung; den Einsatz von Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten Den Strukturwandel in Industrie und Gesellschaft erkennen und beschreiben. Die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen
Der Schwerpunkt verschiebt sich von der Ereignissteuerung zur Behandlung von *Datenstrukturen*. Dabei können OOA/OOD zu völlig verschiedenen Strukturen, beispielsweise mehrfach verkettete Listen führen. In diesem Beispiel wird der Schwerpunkt zunächst auf den direkten Zugriff gesetzt. Die Strukturen sind stark abhängig von der Problemanalyse, der hier sehr viel Zeit gegeben werden. Besonders wichtig sind hier die *Operationen auf Dateien*.	Entwicklung einer Datenorganisation Ein informatisches Modell gewinnen, objektorientierte Analyse
Falls die Analyse ergeben hat, dass es sinnvoll ist, dass auf einzelne Datensätze mit einer Nummer direkt zugegriffen werden kann, wird die Klasse „Direktzugriffsstruktur" eingeführt. Von dieser abstrakten Klasse werden dann Klassen für Direktzugriff im internen (Feld) und externen (Datei) Speicher des Rechners abgeleitet. Bevor die Algorithmen der nächsten Stufe verwirklicht werden können, benötigt man Objekte für Ein- und Ausgabe. Diese werden durch eine Bibliothek vorgegeben.	Ein informatisches Modell gewinnen; objektorientiertes Design; problembezogene Objekte und ihre Wechselwirkungen spezifizieren Verwenden einer Klassenbibliothek
Anschließend werden *Standardalgorithmen* (sequentielles Suchen, mindestens ein Sortierverfahren, binäres Suchen) behandelt. Wenn sie als Methoden einer Oberklasse von „Feld" und „Datei" implementiert werden, können diese Verfahren von Objekten beider Klassen benutzt werden. Die Algorithmen werden jeweils aufgrund einer nötigen Funktionalität des Programms entwickelt.	Den Algorithmenbegriff verstehen; Effizienzuntersuchungen durchführen; Grenzen von Verfahren abschätzen Objektorientiertes Programmieren
Falls die verwendete Programmiersprache dies anbietet, können für die interne Verwaltung von Daten dynamische Felder genutzt werden. Andernfalls muss eine geeignete *dynamische Datenstruktur* entwickelt werden, etwa der Suchbaum. In diesem Fall müssen zusätzlich Konvertierungsalgorithmen zwischen Suchbaum und Datei entworfen werden. Wenn der Termbaum in der Unterrichtseinheit „Taschenrechner" von einer abstrakten Oberklasse „binärer Baum" abgeleitet wurde, kann diese jetzt als Vorlage für den Suchbaum benutzt werden. Sie kann aber auch als Abstraktion von Suchbaum und Termbaum entwickelt werden.	Objektorientiertes Design; problembezogene Objekte und ihre Wechselwirkungen spezifizieren Ein informatisches Modell gewinnen; Entwicklung einer Datenorganisation
Anhand des gewählten oder eines für diesen Zweck geeigneteren konkreten Beispiels für die Dateiverwaltung lassen sich nun Probleme des *Datenschutzes* und der *Datensicherheit* erörtern. Der Bericht des Landesdatenschutzbeauftragten bietet meistens interessante Beispiele für kontroverse Diskussionen.	Die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen Den Einsatz von Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten

(3) Sequenz „objektorientiert visuell“

Vertiefung des objektorientierten Ansatzes unter Verwendung visueller Werkzeuge

Lernsequenzen	Verknüpfung mit der Obligatorik
1. Erzeugen, Verwalten und Entfernen von Objekten (15-20 Stunden)	
Objektorientierte Programmierung (OOP) ergibt ein System von wieder verwendbaren Klassen und Klassenhierarchien (Tools). Die zahlreichen Beziehungen der Objekte und Klassen untereinander lassen sich auf wenige Grundstrukturen reduzieren, die in den folgenden Lernsequenzen auch dem Entwurf neuer, eigenständiger Klassen zugrunde liegen: *Hat-Beziehung* (Zerlegung), *Ist-Beziehung* (Vererbung), *Kennt-Beziehung* (Verbindung).	Konzepte realisieren und optimieren; Programmierkonzepte verstehen und benutzen
Die Hat-Beziehung konkretisiert die „Besitzverhältnisse“ der visuellen Komponenten: Ein Fenster verwaltet seine Komponenten, das Hauptfenster verwaltet alle weiteren zur Laufzeit erzeugten Fenster.	Formen des Strukturierens einsetzen
Die Analyse bisher verwendeter Objekte und Klassen liefert Grundschemata für den Entwurf neuer, eigenständiger Fensterklassen auf der Grundlage der vorhandenen Komponentenbibliothek. Sie verfügen als Unterklassen über die geerbte Funktionalität.	Lösungskonzept als Denkschema entwickeln und Wechselwirkungen analysieren
Das Erzeugen, Verwalten und Entfernen dieser Objekte zur Laufzeit rückt ins Blickfeld der Betrachtung. Die objektorientierten Entwicklungswerkzeuge unterstützen durch das Konzept der Vererbung nur die Realisierung von Ist-Beziehungen. Für die Realisierung der Hat-Beziehungen müssen dagegen eigenständige Schablonen entwickelt werden.	Sprachkonzepte verstehen und beurteilen; Lösungsstrategien systematisieren
Einige Entwicklungswerkzeuge unterstützen die Verwaltung von Hat-Beziehungen durch interne Komponentenlisten. Die Automatismen des Entwicklungswerkzeugs werden reflektiert, evtl. auch modifiziert.	Lösungskonzepte implementieren und nach vorgegebenen Kriterien bewerten
2. Beziehungen zwischen Objekten und Klassen (25-30 Std.)	
Klassen werden durch ihre Funktionalität bestimmt. Das Konzept der Vererbung gestattet es, die Funktionalität einer geeigneten Oberklasse zu übernehmen, zu modifizieren oder zu erweitern. Neue Unterklassen ergeben sich, indem man • die benötigte Funkionalität herausarbeitet, • eine vorhandene Klasse sucht, die bereits Teile dieser Funktionalität besitzt (Oberklasse), • die besonderen Merkmale durch weitere Attribute, neue oder überschriebene Methoden realisiert (Unterklasse).	Allgemeine Strategien verstehen und anwenden
Somit rückt die *Suche nach einer geeigneten Oberklasse* in den Mittelpunkt. Exemplarisch werden gemeinsame Merkmale (z. B. Reaktion auf Paint- und Mausereignisse) aller Komponenten analysiert und dienen als Vorlage für eigene Entwürfe (OOE). Diese Entwürfe erfordern die Erarbeitung von *Dokumentationstechniken* für Klassenhierarchien, Beziehungen und Nachrichtenflüssen.	Strukturen erkennen und beurteilen; Lösungen dokumentieren
Techniken wie Überschreiben von Methoden, statische und virtuelle Methoden, spätes Binden unterstützen die Entwicklung von visuellen (Komponenten) und nicht-visuellen Unterklassen. Neue Methoden ergeben die Erweiterung der Funktionalität. Überschriebene Methoden führen zu einem veränderten Verhalten gegenüber der Oberklasse (*Polymorphie*).	Sprachkonzepte verstehen
Die Legitimierung der einzelnen Entwurfsentscheidungen gemäß den Prinzipien der OOP ergibt zugleich eine weitere Abgrenzung zu imperativer Programmierung.	Sprachkonzepte analysieren und beurteilen

3. Höhere Datenstrukturen (25-30 Std.)	
Algorithmen mit den zugehörigen Daten werden immer an die verwendeten Klassen in Form von Methoden und Zustandsvariablen gebunden. Datenstrukturen wie Listen und Bäume werden daher als Klassen realisiert. Diese Klassen können nicht auf die mächtige Funktionalität der vorhandenen Klassenbibliothek zugreifen. Überlegungen zum sequentiellen und binären Suchen führen zur Klasse *„Liste“* und *„Baum“*. Diese Klassen können beliebige Objekte als Elemente verwalten.	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren
Das Sortierproblem führt zu den Klassen *„Sortierte Liste“* und *„Suchbaum“*. Binäres Suchen und Sortieren setzen die Vergleichbarkeit der Objekte voraus. Elemente dieser Strukturen werden daher aus einer abstrakten Klasse „Sortierbare Elemente“ abgeleitet.	Lösungskonzepte entwickeln und optimieren
Unter Verwendung solcher abstrakten Klassen lassen sich auch die Klassen „Liste“ und „Baum“ zu Klassenhierarchien erweitern, in die Strukturen wie Keller, Schlange, Binärbaum usw. integriert sind.	Algorithmen beurteilen und Effizienzuntersuchungen durchführen
Weitere Untersuchungen zum Begriff des Algorithmus, der Berechenbarkeit und der Komplexität von Lösungen sind weitgehend unabhängig vom verwendeten Sprachkonzept.	

(4) Sequenz „wissensbasiert“

Vertiefung des wissensbasierten Ansatzes

Lernsequenz	Verknüpfung mit der Obligatorik
Der wissensbasierte Ansatz erlaubt eine Vertiefung verschiedener Aspekte: Es können *lernfähige* wissensbasierte Systeme realisiert werden, indem Prädikate benutzt und entwickelt werden, mit denen zur Laufzeit Fakten und Regeln ergänzt, abgefragt beziehungsweise gelöscht werden. Ergänzt um verfeinerte Verfahren zur heuristischen Suche können damit *Modelle selbstlernernder Expertensysteme* entworfen und getestet werden.	Analyse von Anwendungssituationen; Spezifizieren von problembezogenen Objekten und ihrer Wechselwirkungen
Ein größeres Informations- und Dateiverwaltungssystem kann am Beispiel von Auskunfts- oder Reisebuchungssystemen über eine datenbanktechnische Modellierung auf der Basis von *Entity-Relationship-Diagrammen* aufgebaut, es können interaktive Abfragen ermöglicht, sowie Algorithmen für Standardabfragen entworfen und eine Benutzungsschnittstelle realisiert werden.	Entwurf einer Datenorganisation; Modellieren und Konstruieren
Fortgeschrittene Programmier- und Datenstrukturtechniken zur Behandlung von dynamischen baumartigen Strukturen erschließen sich über *verallgemeinerte Listenkonzepte*. Zur Bearbeitung komplex zusammengesetzter Terme wird der univ-Operator „=..“ in Kombination mit der Wurzel-Knoten-Methode eingesetzt. Damit kann ein Term in eine Liste und eine Liste in einen Term umgewandelt werden. Mit den Listenoperationen können typische Strukturuntersuchungen und Manipulationen an Termen durchgeführt werden. Es können Bäume beliebigen Verzweigungsgrades untersucht und eine Vielzahl von Problemen aus dem Umfeld klassischer dynamischer Datenstrukturen behandelt werden, z. B. Termbäume, Parsebäume oder komplexe Suchverfahren auf Graphen.	Daten und Algorithmen abstrahieren; Lösungskonzepte weiterentwickeln; Entwicklung neuer Werkzeuge; Entwickeln und überprüfen von Lösungskonzepten; Erkennen von Chancen und Risiken; Entwicklung und Anwendung von Standardverfahren; fachübergreifendes Arbeiten; Hinführung zu selbständigem Arbeiten in projektähnlichen Organisationsformen

(5) Sequenz „funktional“

Vertiefung des funktionalen Ansatzes

Neben der Vertiefung der oben dargestellten Unterrichtsgegenstände treten in einer weiteren Spiralwindung folgende Aspekte neu auf:

Lernsequenz	Verknüpfung mit der Obligatorik
Abstraktion mit Funktionen - Vertiefung Eine vertiefende Betrachtung der Abstraktionsmöglichkeiten mit Funktionen stellt heraus, dass *Funktionen wie Daten* von Funktionen entgegengenommen werden und von Ihnen zurückgegeben werden können (Funktionsgeneratoren: Funktionen können von anderen Funktionen unter gewissen je individuellen Anfangsbedingungen generiert werden). Denkbar ist etwa eine Beschäftigung mit der Analysis: Funktionen zum Zeichnen und Erstellen von Wertelisten, Verkettung von Funktionen, Funktionsgeneratoren usw. Darüber hinaus können Ableitungsfunktionen generiert werden durch Näherungsverfahren und durch symbolisches Ableiten, wobei zusätzlich das Problem der Vereinfachung von auftretenden Termen als Erweiterung der Manipulation von Wortlisten aus 11 behandelt werden kann. In allen Fällen werden Funktionen generiert, die wie die ursprünglichen verwendet werden können.	Daten und Algorithmen abstrahieren Abschnitt 3: Funktionen als Daten; Daten als Funktionen; Algorithmen, Sprachkonzepte (Grammatiken, ...)
Abstraktion mit Daten - Vertiefung Komplexere Systeme lassen sich erst dann angemessen beschreiben, wenn isolierte Objekte mit einem „Eigenleben“ (einer eigenen lokalen Umgebung) getrennt nebeneinander bestehen können. Dazu benötigt man das fundamental neue Prinzip der *Wertzuweisung* in einem funktionalen System und damit eine Änderung des Auswertungsschemas (*lexical scoping*). Die Funktionen, die die Objekte darstellen, enthalten eine *lokale Umgebung* mit Daten (*Zustandsvariablen*) und Funktionen, sodass diese Objekte auch auf Nachrichten reagieren können. Die Simulation etwa einer Bank ist nur sinnvoll, wenn mehrere Bankkonten unabhängig voneinander existieren. Sie haben alle unterschiedlichen Kontostand und können auf Nachrichten wie (abheben 100) reagieren, indem sie ihren internen Zustand ändern. Die Konstruktoren für die Konten generieren die lokalen Umgebungen; das Bankkonto selber ist eine Funktion, die je nach Anforderung eine der lokalen Funktionen ausführt. Von einer solchen Sicht komplexer Systeme lässt sich, wenn dies gewollt ist, ein Übergang zum *objektorientierten Paradigma* herstellen, auch ohne das System zu wechseln, wenn das funktionale Programmiersystem einen objektorientierten Aufsatz hat.	Lösungskonzepte in einer Programmiersprache; Abschnitt 2: Wertzuweisung, lexical scoping
Eine Ausweitung ist auch denkbar auf Techniken *der direkten Manipulation von Listen mittels Wertzuweisung*. So lassen sich Strukturen wie *Schlangen, Keller*, u.Ä. als Datenstrukturen leichter implementieren.	Daten und Algorithmen abstrahieren; Abschnitt 2: Allgemeine Datenstrukturen
Theoretische Informatik - Vertiefung Neben der Erstellung von Funktionen zur Simulation gängiger *Automatentypen*, zur Simulation *logischer Schaltungen* und der vielfältigen Anwendungsmöglichkeiten bei der *Manipulation von Wortlisten* auf der Grundlage von *Grammatiken* kann etwa in SCHEME vor allem das didaktische Modell der *Engine* verwendet werden, um Funktionsweise von Maschinen und Betriebssystemen (Multitasking, ...) deutlich zu machen.	Algorithmen, Sprachkonzepte; technische, funktionale und organisatorische Prinzipien

(6) Sequenz „anwendungsorientiert“

Vertiefung des anwendungsorientierten Ansatzes unter Verwendung von Datenbankwerkzeugen

Lernsequenz	Verknüpfung mit der Obligatorik
Speicherstrukturen und Zugriffsmethoden Auf der internen Ebene werden die Daten auf Sekundärspeichern abgelegt. Konkret kann man sich die Daten eines Entitys als Menge von Records vorstellen, die in einer Datei gespeichert sind. Bei der physischen Datenorganisation geht es also um die Abbildung der Entitys auf die internen Sätze, die Einrichtung von Zugriffspfaden zu den internen Sätzen und die Einrichtung von Verbindungen zwischen den internen Sätzen. Bei der Betrachtung werden *Effizienzuntersuchungen* hinsichtlich Speichergröße und Laufzeitverhalten angestellt. Bei der Behandlung der Abbildung der Entitys auf interne Sätze wird auf den Aufbau einer Datei, die Recordstruktur und den Begriff der sequentiellen Speicherung eingegangen werden. Die Bedeutung der früher betrachteten Update-Vorgänge wie Einfügen, Löschen und Verändern von Datensätzen für die Veränderungen in einer Datei müssen deutlich gemacht werden. Dabei wird der Datentyp der *linearen Liste* eingeführt. Bei der Abarbeitung von Selektionen kommt es zu Suchvorgängen, die z. B. mit Hilfe eines Index beschleunigt werden können, es wird die *index-sequentielle Speicherung* eingeführt. Dabei wird auch auf den Unterschied der physischen Sortierung der Daten und der sortierten Ausgabe mit Hilfe eines Index eingegangen werden. Sortierverfahren werden nur kurz angesprochen. Eine weitere Möglichkeit zur Erreichung eines schnellen Zugriffes auf Datensätze stellt die Realisierung der Indizes über *B-Bäume* dar. An dieser Stelle wird der Begriff der Baumstruktur eingeführt. Begriffe wie Wurzel, Knoten und Blatt müssen erarbeitet werden. Bei der Realisierung der B-Bäume stellen die Blätter die Datensätze dar. Eine andere wichtige Technik für einen schnellen direkten Zugriff zu einem festgelegten Record mit Hilfe eines Schlüsselwertes stellt das *Hash-Verfahren* dar. An ausgewählten Beispielen wird das Divisions-Rest-Verfahren behandelt. Alle obigen Verfahren dienen in erster Linie der Organisation des Zugriffs über den Primärschlüssel. Für die Realisierung von Sekundärschlüssel wird zusätzlich noch die *Multiliststruktur* eingeführt. Verbindungen werden über Listen und Kettsätze realisiert. Während die Datenstrukturen mit Hilfe von Abbildungen dargestellt werden, wird zur Beschreibung der Algorithmen eine von einer Programmiersprache unabhängige Sprache verwendet	Daten und Algorithmen abstrahieren: allgemeine Strategien und Standardlösungen kennen lernen und anwenden, problembezogene Objekte und ihre Wechselwirkungen spezifizieren, Entwickeln einer Datenorganisation, Aufdecken der Funktionsweise bekannter Werkzeuge, Funktionen und Komponenten der Systemsoftware kennen lernen und beschreiben

Andere Konzepte von Datenbanken; Verbindungen zur theoretischen und technischen Informatik sowie zum Fach Erziehungswissenschaften Es werden zwei andersartige Konzepte von Datenbanksystemen vorgestellt: *Wissensbasierte* und *Objektdatenbanken*. Bei der Behandlung von wissensbasierten Datenbanken/deduktiven Datenbanken wird zuerst das Konzept der logisch-deklarativen Programmierung eingeführt. Die Einführung kann etwa wie in der Sequenz der Stufe 11 zum wissensbasierten Ansatz erfolgen. Es findet dann jedoch der Vergleich zwischen den bisher bekannten Modellen und dem alternativen Ansatz statt. Außerdem werden die Themenbereiche der *Künstlichen Intelligenz* und der *Expertensysteme* angesprochen. Dabei gibt es gute Möglichkeiten zu Verbindungen zum Unterricht im Fach Erziehungswissenschaften, indem in diesem Fach behandelte Lerntheorien und der Intelligenzbegriff aufgegriffen werden und mit Fähigkeiten von Computern verglichen werden. An dieser Stelle werden dann Bezüge zu Gebieten der theoretischen und technischen Informatik einfließen, wenn man auf automatisierte Lösung von Problemen eingeht und die *Automatentheorie* behandelt. Grundlegende Begriffe im Rahmen von Objektdatenbanken sind *Objekte*, *Methoden*, *Kapselung*, *Klassen* und *Beziehungen*. Dabei wird darauf geachtet, dass der Unterschied zwischen Daten, die mit Hilfe objektorientierter Programmiersprachen verwaltet werden und einer Objektdatenbank, die Objekte mit allen Grundfunktionen einer Datenbank verwaltet, deutlich wird.	Algorithmen und Sprachkonzepte beurteilen Systematiken und Theorien zur Lösung spezifischer Anwendungssituationen Fächerübergreifende und fächerverbindende Sichtweisen
Datenbankprojekte Bei der Erstellung einer größeren Datenbank werden alle Aspekte des bisherigen Unterrichtes wieder aufgegriffen. Die Anwenderprogrammierung aus der Stufe 11 wird derart erweitert, dass in einem Programm verschiedene Sichten aufgrund der verschiedenen Zugriffsrechte vergeben werden. Außerdem werden die *Transaktionskontrolle* und die Verwendung der *Abfragesprache* mit in das Programm eingebaut. Dazu gehört auch die Erstellung von *Makros*. Die Kenntnisse über Zugriffsformen führen zu fundierteren Überlegungen über mögliche Primär- und Sekundarschlüsseln.	Erzeugung eines Gesamtbildes von Informatik durch Anwendungsvielfalt/selbständige und projektorientierte Arbeitsformen; Nutzen fortschreitender Analysetechniken/ typische Einsatzbereiche, Möglichkeiten und Grenzen, Chancen und Risiken der Informations- und Kommunikationssysteme untersuchen und einschätzen; Lösungskonzepte in einer Programmiersprache realisieren, überprüfen und weiterentwickeln

3.4.2.2 Paradigmenunabhängige Sequenzen für die Jahrgangsstufen 12 und 13

Für den nicht paradigmatisch gebundenen Teil der Qualifikationsphase wurde in der tabellarischen Übersicht auf maschinennahe Konzepte, Netzstrukturen und ausgewählte Gebiete der theoretischen Informatik abgehoben. Diese Schwerpunktthemen werden im Folgenden eine Stufe weiter ausdifferenziert und dabei wieder an den diaktischen Würfel angebunden.

Dass bei der Darstellung auf ein gröberes Raster zurückgegriffen wird, hat bei allen drei Bereichen einen unterschiedlichen Grund:

Die maschinennahen Konzepte sind bereits durch die landesweiten Lehrerfortbildungsmaßnahmen flächendeckend vorgestellt und durch umfangreiche Materialien aufgearbeitet. Im

Detail kann sich der Unterrichtende also bei seiner Unterrichtsplanung auf umfangreiches und fundiertes Material stützen.

Wer seinen Schwerpunkt auf formal abstrakte und mathematisch notierte Modelle setzen möchte, wird ausgewählte Gebiete der theoretischen Informatik zum Gegenstand des Unterrichts machen. Da mehrere Lehrbücher für die unterrichtliche Aufarbeitung dieses klassischen Stoffgebiets zur Verfügung stehen, kann sich der Lehrplan durch Stichwortnennungen auf die wesentlichen Aspekte beschränken.

Anders liegen die Verhältnisse beim Thema Netzstrukturen. Wegen der zunehmenden Wichtigkeit dieses Themenbereichs wird ausdrücklich dazu ermutigt, sich gegebenenfalls auch auf ein schulisch noch wenig erschlossenes Gebiet informatischen Arbeitens einzulassen. Die an jeweils aktuellen Entwicklungen orientierte Aufnahme neuer Themenfelder in den Unterricht bietet insbesondere die Chance einer sukzessiven thematischen Fortschreibung des Lehrplans.

Maschinennahe Konzepte

Lernsequenz	Verknüpfung mit der Obligatorik
Ausgehend von der konkreten Arbeit mit dem Entwicklungswerkzeug *PC* kann man den Blick einerseits auf die informationstechnologische „Umgebung“ des einzelnen Computers in Form seiner lokalen und globalen Vernetzungen lenken (siehe gesonderten Abschnitt) und andererseits die *logisch-technischen Abläufe und Organisations- bzw. Funktionsprinzipien im Inneren der Maschine* fokussieren.	Kommunikations- und Vernetzungsstrukturen; technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen;
Abstrahierend von allen technischen Finessen aktueller Systeme wird die *Von-Neumann-Maschine* als elementare konzeptionelle und technische Grundlage der meisten verfügbaren Computer aufgedeckt. Mit kleineren Anleihen an *formale Logik* und *Schalttechnik* lassen sich die *basalen Funktionsprinzipen der programmgesteuerten Informationsspeicherung und -transformation* transparent machen und ggf. sogar technisch realisieren bzw. durch (selbst entwickelte) Computersimulationen veranschaulichen.	Aufbau und Funktionsweise bekannter Werkzeuge; Konzept des Von-Neumann-Rechners; Informationsdarstellung für „Daten“/„Befehle“ auf der Maschinenebene
Durch Festlegung auf eine spezielle Realisierung mit elementarem Befehlssatz definiert man eine (reale oder modellartig verfügbare) *Zielmaschine, auf die sich die bekannten Programmierkonstrukte moderner Hochsprachen und Entwicklungsumgebungen durch (didaktisch gestufte) Compilation abbilden lassen*.	Veranschaulichung informatischer Systeme (ggf. Thema einer Facharbeit)
Wesentliche Arbeitsschritte von *Compilern* lassen sich auf überschaubare *Methoden „technischer Übersetzung“ formal wohldefinierter Sprachen* reduzieren. Dabei ist die *Transformationskaskade* von einer weit verbreiteten Programmiersprache (PASCAL) *bis hinunter auf die Ebene eines (Modell)Computers* in den Materialien zur Lehrerfortbildung Informatik Sek. II detailliert beschrieben.	Funktion der Systemsoftware kennen lernen; Syntaxregeln einer formalen Sprache interpretieren
Wert zu legen ist insbesondere auf die *Reduktion des Hochsprachprogramms* auf eine *zeilenorientierte* Form, auf die *Elementarisierung komplexer arithmetischer und logischer Terme*, die Abbildung der *Datenstrukturen in „atomare“ Informationen* und die Umsetzung folgender Konzepte auf Maschinenebene: *Unterprogramme* ohne und mit *Parameterlisten*, *Funktionsaufrufe* und verschiedene *Call-Techniken*. Dabei wird der Unterricht die *direkte* und die *indirekte Adressierung* kombiniert mit *Indexierung* insbesondere bei *Verschiebbarkeit und Replatzierbarkeit von Code- und Datenräumen* zu thematisieren haben.	Schnittstellen untersuchen; Speicherorganisation und Zugriffstechniken erarbeiten
Wer weiter „abwärts“ steigen möchte, macht die *technischen Realisierungen von Speicherbausteinen und Schaltwerken* zum Gegenstand seines Unterrichts und gelangt fließend in *ein Teilgebiet der technischen bzw. theoretischen Informatik.*	Technische Prinzipien von Hard- und Softwaresystemen
Weiter „aufwärts“ rücken die Prinzipen in den Blick, die für sog. *Betriebssysteme* von Computern bedeutsam sind. Insbesondere die *Synchronisation* von in der Maschine *verteilt ablaufenden Prozessen* und die *Zuteilung von Ressourcen* (Speicher, Prozessorleistung, Rechenzeit, ...) *an konkurrierende Anforderer* lassen sich anhand entsprechender Algorithmen unterrichtlich aufarbeiten.	Organisatorische Prinzipien von Hard- und Softwaresystemen

Ausgewählte Gebiete der theoretischen Informatik

Lernsequenz	Verknüpfung mit der Obligatorik
Im Felde der theoretischen Informatik bieten sich klassische Themenbausteine wie die *Theorie der endlichen Automaten*, der *Schaltnetze und Schaltwerke* oder der *formalen Sprachen* an. Auswahl und Akzentuierung wird man davon abhängig machen, welchen theoretischen Hintergrund man dem bisher Gelernten verleihen will.	
Obwohl sich die unterrichtsrelevanten Teile der theoretischen Informatik notativ und vom Kalkül her in mathematisierter Form präsentieren, müssen sie angebunden bleiben an die Fragestellungen der (Schul)Informatik, die im bisherigen Kursverlauf dominierten. Ziel ist damit, die Algorithmisierung, die Sprachkonzepte und Automatenmodelle aus theoretisch fundierter Perspektive heraus beurteilen und analysieren zu können.	Lernen im Kontext
Damit werden die *formale Fassung elementarer Schaltungen*, ihre Aggregation zu *binären Schaltnetzen* und ihre Dynamisierung in *sequentiellen endlichen Automaten* zu komplexen *binären Schaltwerken* nicht nur zum Anwendungsfeld formaler (Schalt)Logik sondern erhellen die *logisch-technischen Grundlagen* des Systems „*Computer*“, bis zu dessen Grundbausteinen man sich im (vorigen oder alternativen) Kursabschnitt beim Abwärtsschreiten von der Hochsprache zur Maschinenebene bereits vorgearbeitet hat.	Analysieren und Bewerten (speziell Feld 2 von Analysieren und Bewerten)
Den Kalkül zu endlichen Automaten kann man auf einer recht elementaren Ebene zur Analyse und anschließenden Synthese bei der Entwicklung von Simulationen technischer Systeme heranziehen.	Anknüpfung an formale Logik und Schaltalgebra
Auch die in der Reduktionskaskade der sukzessiven Compilation schon sehr genau beleuchtete Ausgangs-Programmiersprache kann zur Wurzel einer Sequenz zur theoretischen Informatik werden: die *Theorie der formalen Sprachen* über ihre *Symbole*, zulässigen *Sätze* und definierenden *Grammatiken* beleuchtet die grundsätzliche Frage, welcher Typ von Sprachen *von endlichen Automaten erkannt* werden kann. Dabei erhält die *Syntaxanalyse* eines PASCAL-„Satzes“ bei der Compilation/Interpretation einen einwandfreien theoretisch-formalen Hintergrund. Gleichzeitig wird der Blick dafür geöffnet, mit welchen Einschränkungen die Sätze unserer „*natürlichen Sprache*“ einem „*maschinellem Verstehen*“ zugänglich sind.	Logisch-technische Grundlagen transparent machen; Computersimulation; Konstruktion komplexer technischer Serviceautomaten
Ebenfalls an den Rand des „informatisch Machbaren“ führt eine Sequenz, die sich schwerpunktmäßig mit den *technik-unabhängigen Grenzen der Berechenbarkeit* befasst. Schon in der Klasse der *berechenbaren Funktionen* stößt man auf *beweisbar unlösbare Probleme*. Eine elegante Variante des Berechenbarkeitsproblems ist schließlich *das nicht entscheidbare Halteproblem für die universelle Turingmaschine*, womit ein Grenzgebiet zwischen der Informatik, der Mathematik und der Philosophie angeleuchtet wird (ggf. per Referat).	Formalsprache und Grammatik; Automatenmodelle und akzeptierte Sprachen; Syntaxregeln und Beschreibungssysteme; Problem der Berechenbarkeit; Grenzen von Algorithmisierung; fachübergreifende und fächerverbindende Sichten

Netzstrukturen

Lernsequenz	Verknüpfung mit der Obligatorik
Auf der Basis von elementarem Datenaustausch zwischen zwei Rechnern lassen sich grundsätzliche *Strategien der Kommunikationsorganisation* und ihrer technischen Absicherung verdeutlichen. Möglichkeiten zur Fehlererkennung und -beseitigung durch geeignete Codierungen und Nutzung der Redundanz von Informationen sind unterrichtlich vermittelbar und entweder in Modellen oder aber konkret zwischen PCs an den seriellen oder parallelen Interfaces demonstrierbar.	Handlungsorientierung
Auf eine sehr viel komplexere Ebene begibt sich, wer sich um die Fragen nach *Zugriffsrechten*, *Prioritätsregelungen* und *konfliktfreier Ressourcenvergabe* bei konkurrierenden Anforderungen kümmern will. Dann stellt man zentrale Prinzipien der in Informatikräumen zunehmend häufig anzutreffenden in-house-Vernetzungen mit ihren funktionalen, organisatorischen und administrativen Facetten in den Mittelpunkt.	Aufbau und Struktur lokaler Netzwerke am eigenen System analysieren und beherrschen lernen; Systemadministration; Praxisanbindung
Im Kreise vieler experimentierender Kolleginnen und Kollegen befindet man sich, wenn man *globale Netzwerke* zum Thema des Unterrichts macht: dabei kann die Erstellung von Produkten im WWW bei Einsatz geeigneter Entwicklungswerkzeuge der multimedialen Welt zur Aufgabe des Informatikunterrichts gehören. Die bloße Nutzung im Sinne der fach- oder themenspezifischen Recherche ist hingegen nicht informatikspezifisch, sondern unter medientechnischen und Aktualitätsgesichtspunkten Teil des jeweiligen Fachunterrichts moderner Prägung.	Im Mittelpunkt steht eher der Server als der Surfer; Entwicklungen im WWW anwendungs, produkt- und handlungsorientiert
Eine sorgsame Analyse der Auswirkungen dieser neuen Technologie auf unsere Alltagswelt, auf die Geschäfts- und Berufssphäre und den allgemeinen Wandel der Kommunikationsstrukturen gehört zu den Aufgaben der *Analyse und Bewertung* durch die Informatik und bietet sich für einen kooperativen Unterricht mit Kolleginnen und Kollegen aus anderen Fächern an.	Auswirkungen neuer Informations- und Kommunikationstechnologien analysieren und bewerten; Kooperation mit anderen Fächern

Unterricht im zweiten Halbjahr der Jahrgangsstufe 13

Das letzte Halbjahr des sechssemestrigen Durchgangs wird schon mit Rücksicht auf das geringe Zeitbudget nicht der additiven Erschließung weiterer Themenfelder der Informatik gewidmet, sondern dient in erster Linie der vertiefenden Wiederholung, der Akzentuierung von generalisierenden und reorganisierenden Aspekten des bisher kumulierten Wissens und schließlich der Vernetzung von fachspezifischen und allgemeinbildenen Mosaiksteinen zu einem erweiterten Bild des Unterrichtsfachs Informatik.

Eine von höherer Warte aus angelegte Auffrischung der für das Abiturniveau relevant erscheinenden Sachgebiete unter dem Gesichtspunkt zunehmender inhaltlicher Verquickung und Praxisanbindung mit einem Schwerpunkt auf dem Assessment von Methoden und Produkten kann schließlich dazu beitragen, sowohl die thematische Vielfalt als auch die fachliche Kompetenz und die Fähigkeit zu abgewogener Reflexion informatischen Arbeitens in der Abiturprüfung sichtbar werden zu lassen.

3.5 Mädchen und Jungen im mathematisch-naturwissenschaftlich-technischen Unterricht

Die Verbesserung der Chancengleichheit von Mädchen im mathematisch-naturwissenschaftlich-technischen Unterricht ist Teil eines größeren Problemzusammenhangs. Im mathematisch-naturwissenschaftlich-technischen Unterricht besteht die Gefahr, dass Lehrerinnen und Lehrer ihre Aufmerksamkeit verstärkt den Jungen zuwenden, diese stärker als Individuen ansprechen und ihnen Gelegenheit geben, ihre technisch bestimmten Vorkenntnisse und Interessen im Unterricht zur Geltung zu bringen. Umgekehrt fallen Mädchen wie Jungen im mathematisch-naturwissenschaftlich-technischen Unterricht häufig besonders bereitwillig in geschlechtsstereotypische Verhaltensweisen zurück.

So werden Mädchen oft Zugänge zu naturwissenschaftlich-technischen Unterrichtsfächern erschwert. Sie werden nicht selten demotiviert, erhalten weniger Leistungsanreize und bestätigen dann scheinbar das eigene und das Vorurteil anderer, als Mädchen für solche Fächer nicht begabt zu sein.

Vor diesem Hintergrund muss das Prinzip der Koedukation reflektiert werden. Der mathematisch-naturwissenschaftlich-technische Unterricht soll

- die spezifischen Interessen und Fähigkeiten der Mädchen berücksichtigen und zum Zuge kommen lassen,
- fehlende vor- und außerschulische Erfahrungen von Schülerinnen ausgleichen,
- ein erweitertes Selbstbild bei Mädchen aufbauen helfen,
- zum Abbau von Geschlechtsstereotypen bei Lehrerinnen und Lehrern beitragen.

Dazu ist die didaktische Struktur der Lehrpläne besonders geeignet. Dem „Lernen in Kontexten“ und den „Methoden und Formen des selbständigen Arbeitens“ werden eine größere Bedeutung als in der Vergangenheit zugemessen. Indem das Lernen in Kontexten und die neu akzentuierten Arbeitsformen bewusst auch die spezifischen Arbeits-, Denk- und Frageansätze der Mädchen berücksichtigen, kann erwartet werden, dass ihnen mit den nun vorliegenden Richtlinien und Lehrplänen der gymnasialen Oberstufe neue Zugangsmöglichkeiten zum mathematisch-naturwissenschaftlich-technischen Bereich eröffnet werden. Hierdurch wird auch ein Beitrag zur Verbesserung der Chancengleichheit von Mädchen im Bildungsbereich Mathematik, Naturwissenschaften und Technik geleistet.

Auch wenn sich im Rahmen einer reflexiven Koedukation eine zeitweise Trennung der Geschlechter in bestimmten Fächern bzw. Kursen als förderlich für Mädchen bzw. Jungen erweisen kann, soll die Koedukation nicht aufgegeben werden. Schulen sollen hingegen selbst entscheiden können, ob und bei welchen Gelegenheiten Mädchen und Jungen zeitweise getrennt unterrichtet und wieder zusammengeführt werden.

4. Lernerfolgsüberprüfungen

4.1 Grundsätze

Die Grundsätze der Leistungsbewertung ergeben sich aus den entsprechenden Bestimmungen der Allgemeinen Schulordnung (§§ 21 bis 23). Für das Verfahren der Leistungsbewertung gelten die §§ 13 bis 17 der Verordnung über den Bildungsgang und die Abiturprüfung in der gymnasialen Oberstufe. Die Leistungsbewertung ist Grundlage für die weitere Förderung der Schülerinnen und Schüler, für ihre Beratung und die Beratung der Erziehungsberechtigten sowie für Schullaufbahnentscheidungen. Folgende Grundsätze der Leistungsbewertung sind festzuhalten:

- Leistungsbewertungen sind ein kontinuierlicher Prozess. Bewertet werden alle von Schülerinnen und Schülern im Zusammenhang mit dem Unterricht erbrachten Leistungen (vgl. Kapitel 4.2 und 4.3).
- Die Leistungsbewertung bezieht sich auf die im Unterricht vermittelten Kenntnisse, Fähigkeiten und Fertigkeiten. Die Unterrichtsziele, -gegenstände und die methodischen Verfahren, die von den Schülerinnen und Schülern erreicht bzw. beherrscht werden sollen, sind in den Kapiteln 1 bis 3 dargestellt.

Leistungsbewertung setzt voraus, dass die Schülerinnen und Schüler im Unterricht Gelegenheit hatten, die entsprechenden Anforderungen in Umfang und Anspruch kennen zu lernen und sich auf diese vorzubereiten. Die Lehrerin bzw. der Lehrer muss ihnen hinreichend Gelegenheit geben, die geforderten Leistungen auch zu erbringen.

- Bewertet werden der Umfang der Kenntnisse, die methodische Selbstständigkeit in ihrer Anwendung sowie die sachgemäße schriftliche und mündliche Darstellung. Bei der schriftlichen und mündlichen Darstellung ist in allen Fächern auf sachliche und sprachliche Richtigkeit, auf fachsprachliche Korrektheit, auf gedankliche Klarheit und auf eine der Aufgabenstellung angemessene Ausdrucksweise zu achten. Verstöße gegen die sprachliche Richtigkeit in der deutschen Sprache werden nach § 13 (6) APO-GOSt bewertet.
  Bei Gruppenarbeiten muss die jeweils individuelle Schülerleistung bewertbar sein.
- Die Bewertung ihrer Leistungen muss den Schülerinnen und Schülern auch im Vergleich mit den Mitschülerinnen und Mitschülern transparent sein.
- Im Sinne der Qualitätsentwicklung und Qualitätssicherung sollen die Fachlehrerinnen und Fachlehrer ihre Bewertungsmaßstäbe untereinander offen legen, exemplarisch korrigierte Arbeiten besprechen und gemeinsam abgestimmte Klausur- und Abituraufgaben stellen.
- Die Anforderungen orientieren sich an den im Kapitel 5 genannten Anforderungsbereichen.

4.2 Beurteilungsbereich „Klausuren“

4.2.1 Allgemeine Hinweise

Klausuren dienen der schriftlichen Überprüfung der Lernergebnisse in einem Kursabschnitt. Klausuren sollen darüber Aufschluss geben, inwieweit im laufenden Kursabschnitt gesetzte Ziele erreicht worden sind. Sie bereiten auf die komplexen Anforderungen in der Abiturprüfung vor.

Wird statt einer Klausur eine Facharbeit geschrieben, tritt die Note für die Facharbeit an die Stelle der einer Klausur. Zahl und Dauer der in der gymnasialen Oberstufe zu schreibenden Klausuren gehen aus der APO-GOSt hervor.

4.2.2 Fachspezifische Hinweise zur Aufgabenstellung, Korrektur und Bewertung von Klausuren/Facharbeiten

4.2.2.1 Grundsätze zur Konstruktion von Aufgabenarten

Für die Konstruktion der Aufgaben bzw. Teilaufgaben einer Klausur kommen in erster Linie die folgenden Aufgabenarten in Betracht:

- Analyse und Eingrenzung einer Problemstellung, Entwurf und Spezifikation einer Problemlösung (ggf. unter Zuhilfenahme vorgegebener Werkzeuge) mit erläuternder Begründung sowie einer Beurteilung durch Vergleich alternativer Entwürfe.
- Entwurf einer Problemlösung zu einem spezifizierten Problem, sowie deren Realisierung auch unter Zuhilfenahme vorgegebener Werkzeuge wie z. B. Programmierumgebungen, Modellumgebungen, Klassenbeschreibungen, Allgemeine Datentypen, Prozedurdefinitionen, Prädikatsdefinitionen u.Ä.
- Analyse und Beurteilung einer vorgegebenen Problemlösung/Teillösung zur
  - Ermittlung einer Spezifikation
  - Abschätzung des Aufwandes
  - Herleitung von Aussagen über die Zuverlässigkeit der mit seiner Hilfe erhaltenen Resultate
  - Vergleich mit anderen Realisierungen
  - Einschätzung der Angemessenheit der Problemlösung.
- Vervollständigung, Variation oder Verallgemeinerung eines vorgegebenen Entwurfs oder einer Problemlösung.
- Analyse und Beurteilung von Materialien zum
  - Einsatzbereich von Informations- und Kommunikationssystemen
  - Strukturwandel in Industrie und Gesellschaft
  - ethisch-politischen Umfeld.
- Funktionale Darstellung von Hard- und Softwarekomponenten (im Anschluss an ein im Unterricht behandeltes Thema)
  - eines Netzwerkes oder eines Rechnersystems
  - Vergleiche mehrerer alternativer Rechnersysteme oder Komponenten unter organisatorischen oder algorithmischen Gesichtspunkten.

Zu den genannten Aufgabenarten können in einigen Unterrichtsabschnitten Aufgabenstellungen treten, die den Umgang mit der eingeführten Programmiersprache und den zweckdienlichen Einsatz des verfügbaren Systems überprüfen.

4.2.2.2 Planung von Klausuren

Eine Klausur soll überprüfen, ob die Lernziele eines Kursabschnittes erreicht wurden. Die Lernziele ergeben sich aus einer Synthese der fachlichen Inhalte des Informatikunterrichts (siehe Kapitel 2.1.1) und den Themen, Unterthemen und Unterrichtsgegenständen des ent-

sprechenden Kursabschnitts (siehe Kapitel 3.4 mit den exemplarischen Sequenzbildungen). Bei der Planung der Klausur sollte ein möglichst breites Spektrum schriftlich überprüfbarer Ziele angestrebt werden.

Als allgemeines Prinzip für die Konstruktion von Klausuren ist zu beachten, die einzelnen Aufgaben und Teilaufgaben nach wachsendem Schwierigkeitsgrad anzuordnen. Dabei sollen sich die Anforderungen auf Inhalte und Verfahren beziehen, die im Unterricht behandelt worden sind. Es können dabei auch für die Schülerinnen und Schüler neuartige Teilaufgaben vorkommen, sofern sie mit eingeübten Verfahren zu bearbeiten sind. Diese Anforderungskomponente sollte allerdings keinen dominierenden Anteil an der insgesamt zu erbringenden Leistung ausmachen. Abzulehnen ist eine Aufgabenstellung, bei der gleich zu Beginn ein nicht nahe liegender Lösungsansatz verlangt wird. Eine Anforderung dieser Art, die dem kreativen Bereich zuzuordnen ist, sollte daher allenfalls am Ende einer Aufgabe als zusätzlicher Aufgabenteil gestellt werden.

Die Formulierung einer Aufgabe soll Art und Umfang der geforderten Leistung genau und für die Schülerinnen und Schüler verständlich beschreiben. Komplexe Aufgaben werden in Teilaufgaben gegliedert. Dabei sollte trotz thematischen Zusammenhanges jede Teilaufgabe möglichst unabhängig vom Ergebnis vorhergehender Teilaufgaben behandelt werden können. Eine zu weit gehende Gliederung in Teilaufgaben kann allerdings das Anforderungsniveau mindern sowie kreative und originelle Lösungen erschweren oder verhindern.

Die in der Jahrgangsstufe 11 zu schreibenden Klausuren werden zunächst kürzere Aufgaben zur Kontrolle des Gelernten haben, wobei Gesichtspunkte höherer Leistungsanforderungen noch keine tragende Rolle spielen. Spätestens in der Qualifikationsphase nähern sich Aufbau und Anforderungen einer Klausur den Kriterien, die im Kapitel 5 zur Abiturprüfung im Einzelnen angeführt sind.

Wird der Unterricht im Fach Informatik phasenweise in einzelnen Projektgruppen durchgeführt, so ist bei der Planung von Klausuren zu beachten, dass für verschiedene Projektgruppen u.U. eine gemeinsame Basis der Lerninhalte aus der unmittelbar vorangegangenen Unterrichtszeit fehlen kann. Hier empfehlen sich folgende Maßnahmen:

- In den Klausuren lassen sich nur Aufgaben stellen, die zu keinem der in den einzelnen Gruppen bearbeiteten Teilprojekte in enger inhaltlicher Beziehung stehen. Vielmehr sollen sie die Kenntnis und die richtige Anwendung übergeordneter Begriffe und/oder Problemlösungsverfahren prüfen, die im Verlauf der Projektarbeit erarbeitet oder eingesetzt wurden.
- Sollen Aufgaben gestellt werden, die engere inhaltliche Beziehungen zu den durchgeführten Projekten besitzen, bietet es sich an, die Aufgabenstellung weniger auf die Inhalte als auf den gemeinsamen thematischen Rahmen zu beziehen. Dies bedeutet allerdings, dass die Projektergebnisse (bzw. Teilergebnisse) rechtzeitig vor der Klausur im Plenum eingebracht und besprochen sein müssen.
- Gehören alle Schülerinnen und Schüler, die Klausuren zu schreiben haben, einer Projektgruppe an, so können Aufgabenstellungen auf die Thematik und die Arbeitsergebnisse dieser Projektgruppe zurückgreifen.
- Werden für Schülerinnen und Schüler verschiedener Projektgruppen unterschiedliche Aufgaben gestellt, ist darauf zu achten, dass die Aufgabenstellungen hinsichtlich der Anforderungen vergleichbar sind.

In allen Jahrgangsstufen ist bei Klausuren die Nutzung der verfügbaren Rechneranlage zur Lösung von Klausuraufgaben oder Aufgabenteilen grundsätzlich denkbar und unter den folgenden Bedingungen zulässig:

- Eine gegenseitige Behinderung oder Begünstigung der Schülerinnen und Schüler bei der Nutzung der Hardware muss ausgeschlossen sein. Insbesondere muss eine hinreichend große Anzahl von Arbeitsplätzen zur Verfügung stehen, sodass alle Schülerinnen und Schüler, die an der Klausur teilnehmen, ohne zeitlichen Verzug am Rechner arbeiten können.
- Den Schülerinnen und Schülern dürfen während der Klausur nur die in der Aufgabenstellung vorgesehenen Daten zur Verfügung stehen. Zur Vermeidung von Täuschungsversuchen darf keine Schülerin bzw. kein Schüler Zugang zu Ergebnissen anderer Schülerinnen und Schüler bzw. Zugriff auf sonstige Hilfen haben, soweit diese nicht zur Lösung der Aufgabenstellung ausdrücklich vorgesehen sind.

Den Schülerinnen und Schülern darf bei eventuell auftretenden Funktionsstörungen der Geräte kein Nachteil entstehen. Insofern geht der Kurslehrer ein gewisses Risiko ein, die Klausur im schlimmsten Fall neu stellen zu müssen.

4.2.2.3 Korrektur, Bewertung und Rückgabe von Klausuren

Bei der Korrektur sind auftretende Fehler und Unstimmigkeiten nach Art und Schwere zu kennzeichnen.

Zur Erhöhung der Transparenz und als Lernhilfe für die Schülerinnen und Schüler sollten dort, wo es möglich und sinnvoll ist, erläuternde Hinweise gegeben werden. Besondere, positive Leistungen sollten durch entsprechende Anmerkungen herausgehoben werden. Bei der abschließenden Beurteilung ist der Wert der einzelnen Klausurteile für die Gesamtleistung zu bestimmen.

Zur Transparenz der Leistungsbewertung kann die Einführung von Bewertungseinheiten (Punkten) beitragen, wobei jedoch der Hilfscharakter eines solchen Bewertungsverfahrens zu betonen ist und die Gesamtnote nicht unbedingt und unflexibel aus der erreichten Prozentzahl bestimmt werden muss. Jede klausurinterne Punktwertung ist eine Setzung der Lehrerin bzw. des Lehrers und damit eine subjektive Basis des Beurteilungsverfahrens. Den beurteilenden Fachlehrerinnen und Fachlehrern steht dabei ein Ermessensspielraum zur Verfügung, den sie aufgrund ihrer Erfahrungen und pädagogischen Verantwortung verwenden. Kriterien für die Bewertung einzelner Klausurteile sind z. B.:

- Bedeutung der überprüften Lernziele
- Anteil der erwarteten Arbeitszeit
- Umfang und Komplexität der einzubringenden Teilleistungen
- Grad der geforderten Selbstständigkeit
- Art und Form der Darstellung.

Bei der Festlegung der klausurinternen Punktwertung ist darauf zu achten, dass Schülerinnen und Schüler durch Leistungen mit vorwiegend wiederholendem Charakter eine ausreichende Bewertung erzielen können müssen. Problemlösende und kreative Leistungsanteile führen in der Gesamtwertung höchstens zu einer Differenzierung zwischen einer guten und sehr guten

Leistung. Bei originellen Lösungen, die über die Erwartungen hinausgehen, wird die Lehrerin bzw. der Lehrer das Punktschema durchbrechen, um die Leistung angemessen zu berücksichtigen.

Eine Klausur ist mit ausreichend zu bewerten, wenn annähernd die Hälfte (mindestens vier Zehntel) der erwarteten Gesamtleistung erbracht worden ist. Oberhalb dieser Schwelle sollte die Zuordnung der Punktzahlen zu den höheren Notenstufen von einer linearen Verteilung nicht wesentlich abweichen, desgleichen nicht die Zuordnung zu den beiden Notenstufen unterhalb dieser Schwelle. Die Festlegung dieser Grenze muss bei der Planung der Klausur und der Punktwertung berücksichtigt werden. Die klausurinterne Punktwertung ist den Schülerinnen und Schülern bekannt zu machen.

Bei der Korrektur der Klausur sollten die üblichen Korrekturzeichen für Texte verwendet werden, zusätzlich werden sie durch fachspezifische Korrekturzeichen zu ergänzt.

4.2.2.4 Korrektur und Bewertung von Facharbeiten

Die Lehrerinnen und Lehrer besprechen die Kriterien der Bewertung von Facharbeiten rechtzeitig vor dem Beginn der Erarbeitung mit ihren Schülerinnen und Schülern (siehe auch Kapitel 3.2.2.2). Bei der Bewertung sind u.a. die folgenden Aspekte einzubeziehen:

- Form und Aufbau, z. B. die äußere Form und sprachliche Korrektheit, richtiges Zitieren, Gliederung und gedankliche Strukturierung, ...
- inhaltliches Verständnis, z. B. Erfassen der Aufgabenstellung, Entwicklung einer Lösungsstrategie, Darlegung des Lösungsweges, Formulierung, Diskussion und Bewertung der Ergebnisse, ...
- methodisches Verständnis, z. B. Gestaltung des Arbeitsprozesses, Nutzung der Fachsprache, fachliche Methodenwahl und Methodenbewusstsein, Nutzung von Darstellungsmöglichkeiten und Medien, ...

Die Lehrkraft korrigiert die Facharbeit vor dem Ende des jeweiligen Halbjahres, bewertet sie in einem kurzen Gutachten, das die verschiedenen Aspekte würdigt, erteilt eine Leistungsnote und erläutert ggf. ihre Bewertung.

4.3 Beurteilungsbereich „sonstige Mitarbeit“

4.3.1 Allgemeine Hinweise

Dem Beurteilungsbereich „sonstige Mitarbeit“ kommt der gleiche Stellenwert zu wie dem Beurteilungsbereich Klausuren. Im Beurteilungsbereich „sonstige Mitarbeit“ sind alle Leistungen zu werten, die eine Schülerin bzw. ein Schüler im Unterricht außerhalb der Klausuren erbringt.

Dazu gehören Beiträge zum Unterrichtsgespräch, die Leistungen in Hausaufgaben, Referaten, Protokollen, sonstigen Präsentationsleistungen, die Mitarbeit in Projekten und Arbeitsbeiträge, die in Kapitel 3.2.2 beschrieben sind.

Eine Form der „sonstigen Mitarbeit“ ist die schriftliche Übung, die benotet wird. Die Aufgabenstellung muss sich unmittelbar aus dem Unterricht ergeben. Sie muss so begrenzt sein, dass für ihre Bearbeitung in der Regel 30 Minuten, höchstens 45 Minuten erforderlich sind. Die Schülerinnen und Schüler sollen im Bereich der sonstigen Mitarbeit auf die mündliche Prüfung und deren Anforderungen vorbereitet werden.

4.3.2 Anforderungen und Kriterien zur Beurteilung der Leistungen im Beurteilungsbereich „sonstige Mitarbeit“

Hier werden die einzelnen Arbeitsformen, wie sie bereits in Kapitel 3.2.2 allgemeiner beschrieben sind, in den für die Informatik typischen Bewertungsaspekten beleuchtet.

Beiträge zum Unterrichtsgespräch

Bei der gemeinsamen Erörterung informatischer Sachverhalte kommt es auf die Kenntnis der Fachsprache und auf die Fähigkeit an, Algorithmen und komplexere Strukturen - z. B. Such- und Sortierverfahren, statische und dynamische Datenstrukturen - anschaulich zu erläutern und geeignet formalisiert darzustellen.

Manche Schülerinnen und Schüler sind nicht in der Lage, sich spontan am Unterrichtsgespräch zu beteiligen; ihnen muss die Gelegenheit gegeben werden, an anderer Stelle ihre Fähigkeiten nachzuweisen. Dies ist im Fach Informatik besonders gut möglich, hier können auch die „stillen“ Schülerinnen und Schüler ihre Leistungsfähigkeit aufzeigen, z. B. in der selbständigen Arbeit des Problemlösens als Hausaufgabe.

Hausaufgaben

Im Fach Informatik bieten Hausaufgaben Gelegenheit zu einer selbständigen Auseinandersetzung mit begrenzten neuen Aufgaben. Sie tragen damit dazu bei, dass die Schülerinnen und Schüler fähig werden, Lernvorgänge selbst zu organisieren sowie Arbeitstechniken und Arbeitsmittel selbst zu wählen und einzusetzen.

Häufig wird vor allem die Umsetzung von Problemlösungen auf den Rechner die Hauptaufgabe für die Schülerinnen und Schüler darstellen. Da es nicht Sinn sein kann, die Unterrichtszeit zu einem großen Teil für die Eingabe und Testung von Programmteilen zu verwenden, sollten die Schülerinnen und Schüler Gelegenheit haben, außerhalb der Unterrichtszeit an den Schulrechnern arbeiten zu können, sofern keine anderen Möglichkeiten existieren.

Neben kurzfristigen Hausaufgaben lassen sich auch längerfristige stellen, z. B. zur Entwicklung von Algorithmen und Erstellung von Programmen, für die ein längerer Zeitraum benötigt wird. Es besteht jedoch leicht die Gefahr, im Fach Informatik mehr in der häuslichen Arbeit zu verlangen als dies in anderen Fächern üblich und angemessen ist.

In der gemeinsamen Besprechung einer Hausaufgabe sollten besondere Leistungen hervorgehoben werden; schwächeren Schülerinnen und Schülern wird dadurch geholfen, ihre Fehler zu sehen. Es bedeutet auf jeden Fall eine wichtige Übung für alle Schülerinnen und Schüler, Gedankengänge zu einer Problemlösung vorzutragen bzw. nachzuvollziehen und entscheidende Stellen kritisch zu kommentieren. Verschiedene Lösungsvorschläge sollte man von Schülerinnen und Schülern referieren und die Unterschiede (z. B. Aufwand, Verallgemeinerbarkeit) diskutieren lassen.

Mitarbeit im Team

Das gemeinsame Arbeiten an einem Thema bzw. einer Aufgabe soll in besonderer Weise das soziale Lernen fördern: nicht die einzelne Schülerin bzw. der einzelne Schüler steht im Mittelpunkt, vielmehr ist der gemeinsame Lernerfolg in der Gruppe das Ziel. In der späteren Berufswirklichkeit wird gerade diese Art des Arbeitens und Lernens sehr viel häufiger gefordert als das individuelle Sich-Aneignen unbekannter Sachverhalte und die Einzelarbeit. Insofern kommt dieser Lern- und Arbeitsform eine hohe Bedeutung zu. Die Beurteilung von Schülerleistungen im Team muss neben dem Gesamtergebnis der Gruppenarbeit berücksichtigen,

- wie die Gruppe ihre Aufgaben untereinander aufteilt und Einzelbeiträge sachgerecht integriert,
- wie sie die Kontrolle über die einzelnen Arbeitsabschnitte und Teilleistungen organisiert,
- wie sie sicherstellt, dass jedes Gruppenmitglied zu jeder Zeit über den gesamten Arbeitsfortschritt informiert ist
- und wie sachliche und persönliche Differenzen in der Gruppe überwunden werden.

Umgang mit den verfügbaren Systemen

Die in Schulen verfügbaren Einzelrechner bzw. Rechnernetze sind zusammen mit der Software leistungsfähige und komplexe Systeme, deren Bedienung und gezielte Anwendung erhebliche Anforderungen zunächst und vor allem an die Schülerinnen und Schüler der Jahrgangsstufe 11 stellen. Zu den wesentlichen Arbeitsmethoden des Informatikunterrichts gehören das Testen, Korrigieren und Optimieren selbsterstellter Programme auf dem Rechner. Ihr Zweck ist, zu klären, ob ausgedachte Lösungsverfahren korrekt sind, richtig formuliert wurden und effektiv arbeiten. Folgende Kriterien lassen sich für die Bewertung der Arbeit mit den Geräten der Informatik heranziehen:

- mit der Hard- und Software des Rechners und mit anderen Geräten und Hilfsmitteln, z. B. Modellrechnern vertraut sein,
- das Rechnersystem unter Zuhilfenahme zugehöriger Dokumentationen zweckmäßig und zielgerichtet nutzen,
- auf Zustands- und Fehlermeldungen des Rechnersystems angemessen reagieren.

Schriftliche Übungen

In der schriftlichen Übung sollen die Schülerinnen und Schüler eine sich aus dem Unterricht ergebende, hinreichend vorbereitete und zusammenhängende Problemstellung bearbeiten oder

einen Teilaspekt begründend darstellen. Wesentlich sind dabei das exakte Erfassen der gestellten Aufgabe und die Konzentration auf die wesentlichen Gesichtspunkte.

Die Fachlehrerin bzw. der Fachlehrer ist verpflichtet, im Unterricht auf die erforderlichen Arbeitstechniken hinzuführen und die Schülerinnen und Schüler anzuleiten, eine Aufgabe genau zu analysieren und sachgerecht zu gliedern, den Lösungsweg sinnvoll zu organisieren, die wichtigsten Lösungsschritte festzuhalten, fachliche Zusammenhänge schlüssig aufzuzeigen, die Fachsprache und Arbeitsmethoden sicher einzusetzen und die Arbeitszeit sinnvoll einzuteilen.

Referat und Protokolle

Bei der Erstellung von Referaten und dem Vortrag werden folgende Arbeitstechniken erlernt und geübt:

- die Arbeit organisieren und die Methoden reflektieren,
- themenbezogenes Informationsmaterial beschaffen, zusammenstellen, ordnen und auswerten,
- das Referat gliedern,
- Techniken des Referierens: mit Hilfe einer stichwortartigen Gliederung vortragen, adressatenbezogen sprechen und diskutieren, korrekt zitieren,
- den Zeitfaktor berücksichtigen (bei der Vorbereitung des Referats und dem Vortrag).

Im Hinblick auf den Unterricht kann das Referat sowohl vorbereitenden als auch erweiternden Charakter haben, sowohl Hintergrund- als auch Zusatzinformationen bereitstellen. Das Thema muss eindeutig formuliert und so begrenzt sein, dass es in der vorgesehenen Vorbereitungs- und Vortragszeit bewältigt werden kann. Für die Anfertigung des Referats sollte ein Zeitraum von höchstens zwei Wochen ausreichend sein. Die Vortragszeit sollte in der Regel nicht mehr als 10 Minuten betragen.

Für den Unterricht kommen das Verlaufsprotokoll, ein Protokoll des Diskussionsprofils und das Ergebnisprotokoll in Betracht. Das Anfertigen von Stundenprotokollen gehört zu den studienvorbereitenden Arbeitstechniken. Dabei wird besonders das konzentrierte Zuhören, das Erfassen und Zusammenfassen fachlicher Ausführungen geübt. Das Verlaufsprotokoll soll den Gang der Unterrichtsstunde in den wesentlichen Zügen wiedergeben.

Das Protokoll des Diskussionsprofils greift aus dem Verlauf der Unterrichtsstunde diejenigen Beiträge auf, die die Diskussion entscheidend bestimmt haben. Es macht die unterschiedlichen Standpunkte und ihre Begründung deutlich. Das Ergebnisprotokoll verzichtet auf die Wiedergabe des Unterrichtsverlaufs und auf die Darstellung der Diskussion und hält stattdessen die Unterrichtsergebnisse fest. Die für die Protokolle erforderlichen Arbeitstechniken sollten möglichst früh in der Jahrgangsstufe 11 beginnend eingeübt werden.

Beiträge zu Projektarbeiten

Die Mitarbeit in Projekten ist in besonderer Weise dazu geeignet, Lernprozesse selbständig zu planen, zu organisieren und zu steuern. Projektartige Arbeitsformen werden deshalb systema-

tisch von Beginn an geübt und ständig in Intensität und zeitlichem Umfang erweitert, bis in Stufe 13 komplexere Projektthemen über mehrere Wochen hin bearbeitet werden können.

Die im Rahmen eines Projektes zu bearbeitenden Aufgaben sollen so komplexer Natur sein, dass eine einzelne Schülerin bzw. ein einzelner Schüler sie in der zur Verfügung stehenden Zeit nicht allein lösen kann. Daher wird es notwendig, das Projekt an Gruppen zu vergeben; innerhalb der Gruppe ist dann arbeitsteilig vorzugehen. Der Lösungsweg wird gemeinsam nur grob formuliert, es sind Einzelaufgaben zu isolieren, die Modulschnittstellen müssen genau definiert werden und sind von allen Beteiligten zu beachten, damit zum Schluss eine sinnvolle Gesamtlösung entsteht.

Es leuchtet ein, dass selbständiges Hinarbeiten auf die Lösung eines gestellten Problems in der skizzierten Arbeitsweise nur zu erwarten ist, wenn diese Arbeitsmethode hinreichend vorbereitet und geübt wurde. Weiterhin wird man die Schülerinnen und Schüler auch nicht für mehrere Wochen allein arbeiten lassen sondern die Teilgruppen beobachten, an Gruppendiskussionen teilnehmen, bei erkennbaren Irrwegen Hilfen geben und die Schülerinnen und Schüler durch Protokolle, Zwischenberichte im Plenum u.Ä. zwingen, den Prozess ihrer Arbeit zu reflektieren und jederzeit ihren Standort im Gesamtprojekt zu erkennen und der ganzen Gruppe darlegen zu können.

Die Lehrerin bzw. der Lehrer wird die Arbeitsaufteilung nicht der Gruppe allein überlassen sondern sie mit steuern, z. B. erkennbar anspruchsvollere Teilaufgaben an leistungsstärkere Schülerinnen und Schüler delegieren. Sie/Er wird auch darauf achten, dass besonders leistungsfähige Schülerinnen und Schüler nicht die wesentlichen Arbeiten übernehmen und die anderen lediglich als „Mitläuferinnen“ und „Mitläufer“ von ihnen profitieren. Bei der Beurteilung der Projektergebnisse Leistungen sieht sich die Lehrerin bzw. der Lehrer vor mehrere Schwierigkeiten gestellt:

- Da die Arbeit in einer Gruppe geleistet wird, ist die Einzelleistung der beteiligten Schülerinnen und Schüler oft schwer zu trennen und gerecht zu beurteilen.
- Die Organisation der Arbeit und die Informationsbeschaffung wird nur zu einem Teil im Unterricht selbst erbracht. Daher lässt sich dieser Anteil am Gesamtergebnis oft nicht nach seinem Ablauf, sondern erst am Resultat beurteilen.
- Die Arbeit an Lösungswegen stellt hohe Ansprüche an problemlösendes Denken bis in den kreativen Bereich hinein. Nicht alle Schülerinnen und Schüler können solchen Anforderungen genügen; gleichwohl kann die Arbeit als Ganzes gelingen: Es ist oft schwer auszumachen, wer die entscheidenden Ideen dazu beigetragen hat.
- Schülerinnen und Schüler neigen dazu, längerfristig terminierte Arbeiten schleppend anzugehen.
- Schließlich kann es vorkommen, dass ein Projekt von einer Schülergruppe überhaupt nicht erfolgreich abgeschlossen werden kann, zumindest im Sinne eines funktionierenden Softwaresystems. Auch in diesem Fall muss es möglich sein, die Leistung der Mitglieder dieser Gruppe differenziert zu beurteilen.

Zur Beurteilung der Schülerleistungen im Rahmen von Projekten haben sich folgende Verfahrensweisen und Kriterien bewährt:

- Die Lehrerinnen und Lehrer beteiligen sich von Zeit zu Zeit an der Arbeit einer Projektgruppe. Dabei können sie sich durch Fragen, Anregungen etc. ein Bild von der bereits ge-

leisteten Arbeit und von dem Leistungsvermögen der einzelnen Gruppenmitglieder machen.

- Während der Projektphase sollten Mitglieder der einzelnen Gruppen dem gesamten Kurs Bericht erstatten über ihre Arbeit: Problemstellung, Gliederung, Lösungsansätze und -wege, Stand der Arbeit, noch offene Fragen. Jede Schülerin und jeder Schüler sollte wenigstens einmal referieren.
- Als sinnvoll wird die Führung eines Projektheftes erachtet. Darin dokumentieren alle Schülerinnen und Schüler ihren Anteil am Projekt und protokollieren den Fortgang ihrer Arbeit.

Ein Gruppenergebnis wird von der Lehrerin bzw. vom Lehrer mit einer Gesamtnote bewertet. In einem anschließenden gemeinsamen Gespräch in der Gruppe über das Ergebnis, die individuellen Anteile und ggf. über inhaltliche oder arbeitstechnische Probleme bei der Gruppenarbeit kann und soll die Lehrerin bzw. der Lehrer die Note für die einzelnen Mitglieder nachvollziehbar differenzieren.

5. Die Abiturprüfung

5.1 Allgemeine Hinweise

Es ist spezifische Aufgabe der folgenden Regelungen, das Anforderungsniveau für die Prüfungen im Fach zu beschreiben, die Aufgabenstellung zu strukturieren und eine Beurteilung der Prüfungsleistungen nach verständlichen, einsehbaren und vergleichbaren Kriterien zu ermöglichen. Entscheidend für die Vergleichbarkeit der Anforderungen ist die Konstruktion der Prüfungsaufgaben, die durch Beschluss der KMK[1] in allen Bundesländern nach vereinbarten Grundsätzen erfolgen soll. Diese Grundsätze helfen zugleich, die Beurteilung der Prüfungsbedingungen transparent zu machen.

Zu diesen vereinbarten Grundsätzen gehört die Feststellung, dass den Bedingungen einer schulischen Prüfung zur allgemeinen Hochschulreife die bloße Wiedergabe gelernten Wissens ebenso wenig entspricht wie eine Überforderung durch Problemfragen, die von der Schülerin bzw. vom Schüler in der Prüfungssituation nicht angemessen bearbeitet werden kann. Die Schwerpunkte der Anforderungen liegen in der Abiturprüfung in einem Bereich, der mit selbständigem Aussagen, Verarbeiten und Darstellen bekannter Sachverhalte sowie Übertragen des Gelernten auf vergleichbare neue Situationen beschrieben werden kann.

Die Abiturprüfungsanforderungen sollen deshalb in allen Fächern durch drei Anforderungsbereiche strukturiert werden. Es sind dies:

Anforderungsbereich I	(z. B. Wiedergabe von Kenntnissen)
Anforderungsbereich II	(z. B. Anwenden von Kenntnissen)
Anforderungsbereich III	(z. B. Problemlösen und Werten).

Die Anforderungsbereiche sind für die Lehrerinnen und Lehrer als Hilfsmittel für die Aufgabenkonstruktion gedacht. Sie sollen

- den Lehrerinnen und Lehrern unter Berücksichtigung der Unterrichtsinhalte und ihrer Vermittlung eine ausgewogene Aufgabenstellung erleichtern
- den Schülerinnen und Schülern Verständnis für die Aufgabenstellungen im mündlichen und schriftlichen Bereich erleichtern und ihnen Bewertungen durchschaubar machen
- die Herstellung eines Konsenses zwischen den Fachlehrerinnen und Fachlehrern und damit eine größere Vergleichbarkeit der Anforderungen ermöglichen.

5.2 Beschreibung der Anforderungsbereiche

In der Abiturprüfung sollen die Kenntnisse und Fähigkeiten der Schülerinnen und Schüler möglichst differenziert erfasst werden. Hierbei sind die mit den Aufgaben verbundenen Erwartungen drei Anforderungsbereichen bzw. Leistungsniveaus zuzuordnen, die im Folgenden beschrieben sind.

---

[1] Vereinbarung über die einheitlichen Prüfungsanforderungen in der Abiturprüfung, Beschluß der Kultusministerkonferenz vom 01. Juli 1979, i. d. F. vom 01. Dezember 1989

Anforderungsbereich I

Der Anforderungsbereich I umfasst
- die Wiedergabe von Sachverhalten aus einem abgegrenzten Gebiet im gelernten Zusammenhang
- die Beschreibung und Verwendung gelernter und geübter Arbeitstechniken und Verfahrensweisen in einem begrenzten Gebiet und in einem wiederholenden Zusammenhang.

Dazu kann gehören:
- das Wiedergeben grundlegender Begriffe, Definitionen und Aussagen
- das Wiedergeben, Identifizieren und sachgerechte Anwenden algorithmischer Grundstrukturen und elementarer Datentypen
- das Beschreiben bekannter Verfahren, Methoden und Prinzipien der Informatik
- das Verwenden und Bedienen bekannter Geräte der Informatik in eingeübter Arbeitsweise
- das Wiedergeben eines bekannten Grundalgorithmus in einer im Unterricht behandelten Darstellungsform
- die Nutzung von Wertverlaufstabellen zur Darstellung der Werte und Wertänderungen von Objekten, die algorithmisch verändert werden
- das Beschreiben konkreter, im Unterricht behandelter Datenstrukturen
- die Wiedergabe syntaktischer Elemente der im Unterricht benutzten Programmiersprache einschließlich ihrer Bedeutung und der Regeln ihrer Zusammensetzung in der eingeübten Darstellungsweise
- das Übertragen bekannter syntaktischer Strukturen in zugehörige Sequenzen der Programmiersprache nach behandelten Regeln
- das Beschreiben des Anwendungsbereiches oder des Gültigkeitsbereiches von Modellen der Informatik in einem wiederholenden Zusammenhang
- das Beschreiben des Aufbaus und der Funktionsweise bekannter Geräte aus der Informatik und ihrer Komponenten
- das Beschreiben und Verwenden der behandelten maschineninternen Zahlendarstellungen und der Operationen mit ihnen
- das Beschreiben grundlegender Eigenschaften im Unterricht behandelter Datenstrukturen.

Anforderungsbereich II

Der Anforderungsbereich II umfasst
- selbständiges Auswählen, Anordnen, Verarbeiten und Darstellen bekannter Sachverhalte unter vorgegebenen Gesichtspunkten in einem durch Übung bekannten Zusammenhang
- selbständiges Übertragen des Gelernten auf vergleichbare neue Situationen, wobei es entweder um veränderte Fragestellungen oder um veränderte Sachzusammenhänge oder um abgewandelte Verfahrensweisen gehen kann.

Dazu kann gehören:
- die Verwendung bekannter Fakten, Definitionen, Begriffe, Aussagen, Begründungen und Schlussfolgerungen bei der Bearbeitung einer neuen Fragestellung aus einem im Unterricht behandelten Stoffbereich
- die Anwendung bekannter Verfahren, Methoden und Prinzipien der Informatik bei der Lösung einer neuen Aufgabe aus einem bekannten Problemkreis

- das Zusammensetzen formaler Sprachelemente zur syntaktisch und semantisch korrekten Beschreibung einer gegebenen algorithmischen Problemlösung
- die Übertragung eines gegebenen Algorithmus von einer Darstellungsform in eine andere
- die syntaktische und semantische Analyse eines vorgegebenen Programms
- der Nachweis der Endlichkeit eines im Kern bekannten Problemlösungsverfahrens
- die Abschätzung des Zeit- und Speicheraufwandes eines Algorithmus
- das Ersetzen gegebener Kontrollstrukturen oder Datenstrukturen durch äquivalente andere
- der Vergleich vorgegebener oder selbst entworfener Algorithmen und Modelle mit Hilfe bekannter Kriterien und Begriffe
- die sachgerechte Anwendung bekannter Algorithmen in einem neuen Sachzusammenhang
- der Entwurf eines Algorithmus zur Lösung eines schon in ähnlicher Weise behandelten Problems
- die Auswahl einer zur Lösung eines Problems geeigneten Datenstruktur und die Begründung dieser Wahl
- die Begründung der numerischen Stabilität oder Instabilität eines Algorithmus mit Hilfe im Unterricht geübter Methoden.

Anforderungsbereich III

Der Anforderungsbereich III umfasst planmäßiges Verarbeiten komplexer Gegebenheiten mit dem Ziel, zu selbständigen Lösungen, Gestaltungen oder Deutungen, Folgerungen, Begründungen, Wertungen zu gelangen. Dabei werden aus den gelernten Methoden oder Lösungsverfahren die zur Bewältigung der Aufgabe geeigneten selbständig ausgewählt oder einer neuen Problemstellung angepasst.

Dazu kann gehören:
- das systematische Entwickeln eines Algorithmus zur Lösung eines allgemeinen Problems, zu dem im Unterricht höchstens ein Spezialfall behandelt wurde
- das Auffinden eines Lösungsansatzes für ein Problem, wobei Kenntnisse aus verschiedenen Teilgebieten der Informatik verbunden werden müssen, ohne dass dies in vergleichbaren Zusammenhängen geübt wurde
- das Entwerfen und Verwenden problemgerechter Datenstrukturen zur Bearbeitung von Objekten mit anderen als nur den aus dem Unterricht bekannten Eigenschaften
- das Interpretieren von Ergebnissen in nicht vom Unterricht her bekannten Zusammenhängen
- die syntaktische und semantische Analyse eines vorgegebenen Algorithmus zur Lösung einer im Unterricht nicht besprochenen Problemklasse
- das Auffinden, Formulieren oder Begründen einer Vermutung, die sich aus der Bearbeitung einer Teilaufgabe oder dem Vergleich der Ergebnisse mehrerer Teilaufgaben ergibt
- das Ausführen eines Beweises, zu dem eigenständige Beweisgedanken erforderlich sind
- das Ersetzen eines gegebenen instabilen numerischen Verfahrens durch ein selbst zu findendes stabiles Verfahren
- das Entwickeln eines Modells zur Beschreibung oder Simulation bisher nicht bearbeiteter Phänomene
- die Abschätzung und Wertung der Anwendungsmöglichkeiten informatischer Methoden und Gesetzmäßigkeiten im sozialen bzw. politischen Raum.

5.3 Die schriftliche Abiturprüfung

Zur Art der Aufgabenstellung, zur Vorlage der Aufgabenvorschläge bei der oberen Schulaufsichtsbehörde, zur Korrektur und Bewertung der schriftlichen Arbeiten gelten grundsätzlich die §§ 32 bis 34 der APO-GOSt und die entsprechenden Verwaltungsvorschriften.

5.3.1 Aufgabenarten der schriftlichen Abiturprüfung

Ein Vorschlag für die schriftliche Abiturprüfung im Fach Informatik kann im Grundkurs aus einer einzigen oder aus zwei voneinander unabhängigen Aufgaben bestehen. Im Leistungskurs besteht er immer aus zwei Aufgaben.

Durch eine Untergliederung der Aufgabenstellung in Teilaufgaben erreicht man eine Mischung verschiedener Aufgabenarten und kann so die in 5.2 beschriebenen Anforderungsbereiche in ihrer Gesamtheit geeignet gewichtet realisieren. Die Aufgaben erreichen dann ein angemessenes Niveau, wenn der Hauptanteil der zu erbringenden Prüfungsleistungen im Anforderungsbereich II liegt und daneben die anderen beiden Anforderungsbereiche berücksichtigt werden, und zwar Anforderungsbereich I in deutlich höherem Maße als Anforderungsbereich III.

In diesem Zusammenhang ist zu beachten, dass die zum Wesen der Informatik konstitutiv gehörende Suche nach Problemlösungsverfahren erhebliche Anforderungen aus dem Bereich III stellen kann. Um hier - vor allem im Grundkursfach - nicht zu überfordern, sind in jedem Fall Aufgabenstellungen zu vermeiden, zu deren Lösung gleich zu Anfang eine besondere, nicht nahe liegende oder nicht im Unterricht vorbereitete Idee gefunden werden muss. Dann besteht leicht die Gefahr, dass die Prüflinge dieser hohen Einstiegsschwelle wegen die ganze Aufgabe nicht lösen können. Als kreativ einzustufende Anforderungen sollten daher allenfalls am Ende einer Aufgabe gestellt werden.

Eine scharfe Abgrenzung der Anforderungsbereiche gegeneinander ist häufig nicht möglich. Auch die Zuordnung einzelner Aufgabenteile oder Lösungsschritte zu einem bestimmten Bereich lässt sich nicht immer eindeutig vollziehen. Die Art und der Umfang der Vorbereitung im Unterricht sind für die unterschiedliche Einstufung in die Anforderungsbereiche maßgebend.

5.3.1.1 Beispiele für typische Aufgabenarten

Für die schriftliche Abiturprüfung im Fach Informatik sind folgende Aufgabenarten denkbar:

(1) Entwicklung eines Algorithmus zur Lösung einer Problemstellung mit erläuternder Begründung des Lösungsansatzes und Formulierung des Algorithmus in verbaler, graphischer oder programmsprachlicher Darstellung,

(2) Präzisierung und Verfeinerung eines in grober Umschreibung vorgelegten Algorithmus mit Erfassen aller Sonderfälle,

(3) Niederschrift eines Algorithmus in Form eines Programms,

(4) Analyse und Beurteilung eines vorgegebenen Programms/Algorithmus
- zur Ermittlung des Problems, das durch das Programm/den Algorithmus gelöst wird,
- zur Abschätzung des Aufwandes (Zeitkomplexität und/oder Speicherbedarf),
- zur Herleitung von Aussagen über die numerische Zuverlässigkeit der mit seiner Hilfe erhaltenen Resultate.

(5) Entwurf und/oder Beurteilung einer Datenstruktur zur Beschreibung von Objekten und der Relationen zwischen ihnen im Rahmen einer gegebenen Problemstellung,

(6) Bearbeitung einer Aufgabe aus den Bereichen der technischen, der theoretischen oder der angewandten Informatik.

5.3.1.2 Grundsätze zur Konstruktion von Prüfungsaufgaben

Zur Konstruktion von Prüfungsaufgaben sind neben den allgemeinen Anforderungen in der schriftlichen Abiturprüfung folgende informatikfachliche Grundsätze zu beachten:
- Die Prüfungsaufgabe(n) ist bzw. sind im Allgemeinen in mehrere Teilaufgaben zu gliedern. Diese sollen eine strukturierende Bearbeitung der Gesamtaufgabe erleichtern, jedoch nicht zu stark einengen.
- Die Teilaufgaben müssen untereinander in erkennbarem Sachzusammenhang stehen. Sie dürfen jedoch nicht so weit voneinander abhängen, dass nachfolgende Teilaufgaben nur gelöst werden können, wenn die vorhergehenden korrekt bearbeitet wurden.
- Bei allen Aufgaben gehören erläuternde Texte zum Lösungsansatz, Begründungen zu einzelnen Lösungsschritten und eine Diskussion von Ergebnissen zu den Prüfungsforderungen.
- Die Prüflinge müssen alle Prüfungsaufgaben bearbeiten, sie haben keine Auswahlmöglichkeit unter Aufgaben oder Teilaufgaben.
- Fachübergreifende Probleme können nur dann Gegenstand von Prüfungsaufgaben sein, wenn die dazu erforderlichen Kenntnisse trivial sind, der Themenkreis im Unterricht hinreichend behandelt wurde oder die erforderlichen Kenntnisse im Aufgabentext umfassend geklärt werden.
- Aufgaben, die mit Hilfe eines Rechners zu lösen sind, unterliegen den gleichen Bedingungen wie Semesterklausuren (siehe Kapitel 4.2.2.2). Den Prüflingen sollte die besondere Situation von der bisherigen Klausurpraxis her bekannt sein.
- Eine gemeinsame Aufgabenstellung für parallele Kurse, auch an verschiedenen Schulen, ist bei entsprechenden Vermerken in den eingereichten Vorschlägen und entsprechender Organisation der schriftlichen Prüfung zulässig.

5.3.2 Einreichen von Prüfungsvorschlägen

(1) Die Fachlehrerin bzw. der Fachlehrer legt zur Wahl für die Prüflinge zwei Prüfungsvorschläge einschließlich der Genehmigungsunterlagen vor, von denen die obere Schulaufsicht einen Vorschlag ausgewählt. Zur Aufgabenstellung der schriftlichen Abiturprüfung

ist § 33 Abs.1 APO-GOSt zu beachten. Die Aufgabenvorschläge in der schriftlichen Abiturprüfung müssen aus dem Unterricht in der Qualifikationsphase erwachsen sein. Die der Schulaufsicht vorzulegenden Vorschläge müssen sich in ihrer Breite insgesamt auf die Ziele, Problemstellungen, Inhalte und Methoden der vier Halbjahre der Qualifikationsphase beziehen und unterschiedliche Sachgebiete umfassen. Der zu bearbeitende Prüfungsvorschlag muss sich in der Breite der Ziele, Problemstellungen, Inhalte und Methoden mindestens auf zwei Halbjahre der Qualifikationsphase beziehen.

(2) Den Prüfungsvorschlag macht die Fachlehrerin oder der Fachlehrer der Jahrgangsstufe 13/II, gegebenenfalls unter Beteiligung der Kurslehrerinnen und Kurslehrer, von denen die Schülerinnen und Schüler in den Jahrgangsstufen 12 und 13 im Fach unterrichtet worden sind. Fachlehrerinnen und Fachlehrer, die Aufgabenvorschläge für Schülergruppen mit unterschiedlichen Kursvoraussetzungen stellen, legen die Vorschläge für jede Schülergruppe gesondert vor.

(3) Dem Prüfungsvorschlag sind beizufügen
- eine kurz gefasste konkrete Beschreibung der erwarteten Schülerleistung (Erwartungshorizont) unter Hinweis auf die konkreten unterrichtlichen Voraussetzungen
- eine hinreichend detaillierte Angabe über die Lerninhalte der Halbjahreskurse
- die Erklärung der Fachlehrerin oder des Fachlehrers, dass das Notwendige für die Geheimhaltung veranlasst wurde.

(4) Die vorgesehenen Hilfsmittel sind am Schluss eines jeden Vorschlags aufzuführen.

5.3.3 Bewertung der schriftlichen Prüfungsleistungen

Die schriftlichen Abiturprüfungsarbeiten werden grundsätzlich wie Semesterklausuren bewertet (siehe Kapitel 4.2.2.3).

5.3.4 Beispiele für Prüfungsaufgaben in der schriftlichen Abiturprüfung

Die drei folgenden Aufgabenbeispiele stellen jeweils einen vollständigen Abiturvorschlag für einen Grundkurs dar. Aufgabe 1 und Aufgabe 2 zusammen bilden einen Vorschlag für einen Leistungskurs. Gleiches gilt für Aufgabe 1 und Aufgabe 3.

Aufgabe im schriftlichen Abitur (1. Beispiel, programmiersprachunabhängig)

Lernvoraussetzungen

Im Bereich ausgewählter Themen zur theoretischen Informatik ist der endliche Automat formal eingeführt und an einigen Beispielen bearbeitet und eingeübt worden, ebenso Regelsysteme für die von diesen Automaten erkannten Sprachen (rechtslineare Grammatiken). In einem weiteren Unterrichtsabschnitt ist die Reduktion von einer höheren Programmiersprache (hier PASCAL) über eine Zwischensprache auf Assemblerebene (hier der aus der Lehrerfortbildung bekannte ALI) erarbeitet worden. Lerninhalte aus beiden Bereichen werden in der folgenden Aufgabe, die vom Umfang her als **eine** von zwei Aufgaben eines Abiturvorschlages anzusehen ist, angesprochen.

Aufgabenstellung

Gegeben ist ein endlicher Automat $A = (S, \Sigma, s_0, F, \delta)$ mit $\Sigma = \{ 0, 1\}$, $S = \{ s_0, s_1, s_2 \}$, $F = \{ s_0 \}$ und der Übergangsfunktion δ mit $\delta(s_0, 0) = s_0$, $\delta(s_0, 1) = s_1$, $\delta(s_1, 0) = s_2$, $\delta(s_1, 1) = s_0$, $\delta(s_2, 0) = s_1$, $\delta(s_2, 1) = s_2$.

a) Zeichne den zugehörigen Zustandsgraphen!

b) Interpretiere die folgenden Worte als Dualzahlen und prüfe, welche davon durch den gegebenen Automaten akzeptiert werden:

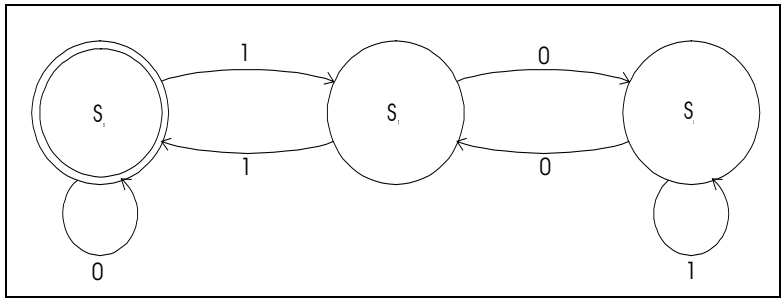
Wort	Dezimalzahl	akzeptiert?	Wort	Dezimalzahl	akzeptiert?
0			1000		
1			1001		
10			1010		
11			1011		
100			1100		
101			1101		
110			1110		
111			1111		

Welche gemeinsame Eigenschaft haben offenbar alle zur Sprache L(A) des Automaten gehörigen und hier als Zahlen interpretierten Worte? Begründe dies anhand des Zustandsgraphen!
(Hinweis: Wie verändert sich der Wert einer Dualzahl durch Anfügen einer 0 bzw. einer 1 und was bedeutet das für die Divisionsreste?)

c) Erläutere, was man unter einer rechtslinearen Grammatik versteht! Gib eine Grammatik an, welche die gleiche Sprache erzeugt, die auch der vorstehende Automat akzeptiert!

d) Erstelle ein ALI-Programm zur Simulation dieses Automaten! Der Benutzer soll dabei eine Folge von Nullen und Einsen eingeben können, wobei die Eingabe der Zahl -1 diese Folge beenden soll. Falls das eingegebene Wort akzeptiert wird, soll das Programm eine 1 ausgeben, andernfalls eine -1.

Musterlösung



a) Zustandsgraph zum Automaten (siehe rechts)

b) Anhand der Tabelle erkennt man, dass offenbar alle durch 3 teilbaren Zahlen in Dualdarstellung vom Automaten akzeptiert werden. Zur Begründung überlegt man sich, dass ein Anfügen einer 1 an eine bestehende Dualzahl einer Multiplikation mit 2 und einer Addition von 1 entspricht; dabei entsteht eine Zahl, die bei der Division den Rest 1 lässt. Ausgehend vom Zustand s_0 befindet man sich dann im Zustand s_1. Ein anschließendes Anfügen einer 0 (Multiplikation mit 2) führt auf eine Zahl, die bei der Division durch 3 den Rest 2 lässt (Zustand s_2). Weiteres Anfügen einer 1 multipliziert diese Zahl dann zuerst wieder mit 2, was einem „Rest von 4“ entspräche, tatsächlich aber modulo 3 gerechnet einen Rest von 1 ergibt, addiert dann aber noch wieder 1 dazu, sodass der Rest letztlich 2 bleibt, der Automat also im Zustand s_2 verharrt. Fügt man - ausgehend von s_2 – dagegen eine 0 an, so führt dies - wie schon gesagt - zu einer Zahl mit dem Rest 1, also wieder in den Zustand s_1 zurück. Fügt man dann eine 1 an, so verdoppelt sich zunächst wieder der Rest auf 2, und durch das anschließende Addieren einer 1 kommt man wieder zu einer durch 3 teilbaren Zahl und damit in den Zustand s_0 (Endzustand) zurück. An eine durch 3 teilbare Zahl dürfen dann noch beliebig viele Nullen (in Dualdarstellung) angehängt werden, was wegen der jeweiligen Verdopplungen an der Teilbarkeit nichts ändert.

c) Die Grammatik $G = (T, N, P, S^0)$ einer formalen Sprache besteht aus einer Menge T von terminalen Symbolen A, B, C, ..., einer Menge N nichtterminaler Symbole a, b, c, ..., einer Menge P von Produktionsregeln und dem Startsymbol S^0. Bei einer rechtslinearen Grammatik kommen dabei als Produktionsregeln nur Regeln der folgenden Form vor: $A \rightarrow aB$, $A \rightarrow B$, $A \rightarrow a$ und $A \rightarrow \varepsilon$. Wählt man als Startsymbol S^0 den Zustand s_0 des

Automaten, $T = \Sigma = \{ 0, 1 \}$, $N = S = \{ s_0, s_1, s_2 \}$, so lässt sich mit dem folgenden Produktionsregelsystem eine rechtslineare Grammatik zu dem hier gegebenen Automaten angeben: $P = \{ s_0 \rightarrow \varepsilon, s_0 \rightarrow 0s_0, s_0 \rightarrow 1s_1, s_1 \rightarrow 0s_2, s_1 \rightarrow 1s_2, s_2 \rightarrow 0s_1, s_2 \rightarrow 1s_2 \}$.

d) ALI-Programm zur Simulation des Automaten:

AUTOMAT	START	0	Programmkopf
S0	INI	EINGABE	Automat im Zustand S0 wartet auf Eingabe von 0, 1 oder -1,
	L	0, EINGABE	lädt diese in den Akku,
	C	0, NULL	vergleicht sie mit 0,
	BE	S0	verzweigt zu S0, falls Eingabe = 0 (bleibt also in S0),
	BL	AKZEPTIERT	verzweigt zu Akzeptiert, falls Eingabe = -1 (Abschlusszeichen),
	B	S1	verzweigt zu S1 sonst (also wenn Eingabe = 1).
S1	INI	EINGABE	Automat im Zustand S1 wartet auf Eingabe von 0, 1 oder -1,
	L	0, EINGABE	lädt diese in den Akku,
	C	0, NULL	vergleicht sie mit 0,
	BE	S2	verzweigt zu S2, falls Eingabe = 0,
	BL	ABGELEHNT	verzweigt zu Abgelehnt, falls Eingabe = -1 (Abschlusszeichen),
	B	S0	verzweigt zu S0 sonst (also wenn Eingabe = 1).
S2	INI	EINGABE	Automat im Zustand S2 wartet auf Eingabe von 0, 1 oder -1,
	L	0, EINGABE	lädt diese in den Akku,
	C	0, NULL	vergleicht sie mit 0,
	BE	S1	verzweigt zu S1, falls Eingabe = 0,
	BL	ABGELEHNT	verzweigt zu Abgelehnt, falls Eingabe = -1 (Abschlusszeichen),
	B	S2	verzweigt zu S2 sonst (also wenn Eingabe = 1 bleibt er in S2).
AKZEPTIERT	OUTI	EINS	Ausgabesymbol für ein akzeptiertes Wort,
	B	ENDE	Sprung zum Ende des Programms,
ABGELEHNT	OUTI	MINUSEINS	Ausgabesymbol für ein nicht akzeptiertes Wort,
ENDE	EOJ		Ende des Programms, es folgt der Deklarationsteil:
EINGABE	DS	F	2 Byte Speicherplatzreservierung für eine Ganzzahl
NULL	DC	„0“	Konstantendeklarationen ...
EINS	DC	„1“	
MINUSEINS	DC	„-1“	
	END	AUTOMAT	Ende.

Aufgabe im schriftlichen Abitur (2. Beispiel, Themenbereich Datenstrukturen - OOP)

Lernvoraussetzungen und Anforderungen

Die Techniken zur Realisierung verketteter Strukturen sind im Unterricht an verschiedenen Beispielen (Listen- und Baumstrukturen) erarbeitet worden. Die hier verlangte selbständige und nicht schematische Darstellung dieser prinzipiell bekannten Sachverhalte ist im Wesentlichen dem Anforderungsbereich II zuzuordnen.

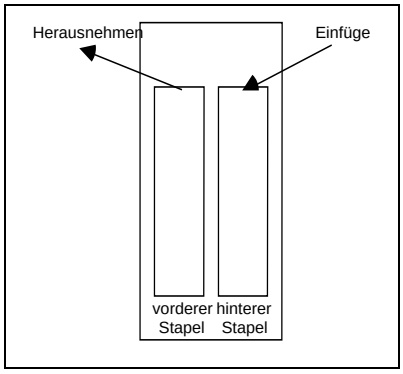
Das Modell des Doppelstapels als mögliche Realisierung einer Schlange ist den Schülerinnen und Schülern nicht bekannt. Natürlich kennen sie aber die kennzeichnenden Eigenschaften (LIFO, FIFO) der Strukturen Stapel und Schlange. Diese Kenntnisse müssen hier auf eine neue Situation angewandt werden, wobei der nicht angegebene Zeitpunkt des Umschaufelns nur eine kleine Hürde bei der Lösung der Aufgabe darstellen dürfte, da kaum andere als der richtige Zeitpunkt möglich sind (Anforderungsbereich II).

Bei der Realisierung in PASCAL können die Schülerinnen und Schüler auf viele geübte Techniken der verwendeten Programmiersprache zurückgreifen. Die Verwendung von vorgegebenen Dokumentationen bei der Realisierung neuer Klassen ist ebenfalls eine häufig benutzte Technik. Die neuen Sachzusammenhänge müssen hier jedoch durch selbständiges Übertragen des Gelernten gelöst werden. Die Aufgabe ist somit dem Anforderungsbereich II zuzuordnen.

Aufgabenstellung

a) Die Klasse Stapel wurde von uns in zwei unterschiedlichen Versionen mit Hilfe der Verkettungs- und der Klammertechnik realisiert. Erläutere die beiden Techniken für den Fall des Stapels und beschreibe insbesondere die Vorteile der Klammertechnik.

b) Das folgende Schaubild beschreibt eine Datenstruktur, die aus zwei Stapeln besteht:
Beim Einfügen wird ein neues Element stets auf den hinteren Stapel gelegt. Herausgenommen werden die Elemente **nur** aus dem vorderen Stapel. Dazu müssen natürlich Elemente vom hinteren zum vorderen Stapel gelangen. Dies geschieht durch so genanntes **Umschaufeln**, wobei alle Elemente aus dem hinteren Stapel nacheinander entfernt und auf den vorderen Stapel gelegt werden.
Zeige, dass die hier beschriebene Struktur des Doppelstapels eine Schlange ist, wenn man die Stapel in einem geeigneten Moment umschaufelt. Wann ist dieser geeignete Moment?
Erläutere die Idee anhand des folgenden Beispiels durch Momentaufnahmen im obigen Modell:



- die Elemente A,B,C werden eingefügt
- die Elemente D,E,F,G werden eingefügt
- H wird eingefügt
- zwei Elemente werden herausgenommen
- drei Elemente werden herausgenommen
- drei Elemente werden herausgenommen

c) Realisiere die Klasse Schlange (siehe Anlage) nach dem in b) beschriebenen Modell in Pascal. Dabei kann davon ausgegangen werden, dass eine Realisierung der Klasse Stapel (siehe Anlage) in einer Unit mStapel bereits vorliegt.
Welche Klassen- bzw. Objektstruktur zwischen den Klassen Stapel und Schlange ergibt sich bei dieser Realisierung?
Hinweis: Realisiere gesondert eine Methode Umschaufeln. Sollte deren Realisierung nicht gelingen, kann im weiteren Verlauf dennoch auf diesen Dienst zurückgegriffen werden.

Anlage 1

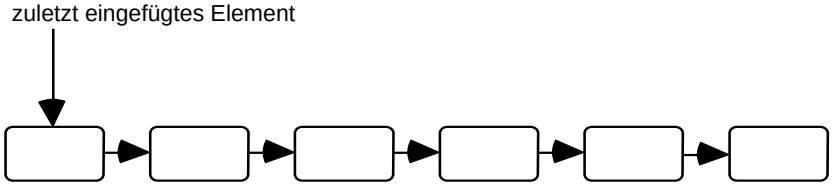
Klasse:	Stapel
Oberklasse:	
Beschreibung:	Ein „Stapel“ verwaltet beliebige Elemente (Klasse Basis) nach dem LIFO-Prinzip, d.h. das Element, das als Letztes in den Stapel eingefügt wurde, wird als Erstes wieder entfernt. Nur auf das zuletzt eingefügte, aktuelle Element kann zugegriffen werden.
Bezugsklasse:	Basis
Unterklassen:	
Protokoll:	
Auftrag	IStapel
nachher	Der Stapel ist initialisiert. Er enthält keine Elemente.
Auftrag	Einfuegen (neues: Basis)
nachher	Das Element „neues“ befindet sich auf dem Stapel. Es ist das zuletzt eingefügte.
Auftrag	Loeschen
vorher	Der Stapel ist nicht leer.
nachher	Das zuletzt eingefügte Element ist aus dem Stapel entfernt.
Anfrage	Leer: Wahrheitswert
nachher	Diese Anfrage ist wahr, wenn der Stapel leer ist.
Anfrage	ZeigeElement: Basis
vorher	Der Stapel ist nicht leer.
nachher	ZeigeElement ist das zuletzt hinzugefügte Element des Stapels.
Auftrag	GibFrei
nachher	Der Stapel existiert nicht mehr.

Anlage 2

Klasse:	Schlange
Oberklasse:	
Beschreibung:	Eine Schlange verwaltet beliebige Elemente (Klasse Basis) nach dem FIFO-Prinzip, d.h. das Objekt, das als Erstes in die Schlange eingefügt wurde, wird als Erstes aus der Schlange entfernt. Es können beliebig viele Elemente in der Schlange enthalten sein.
Bezugsklasse:	Basis
Unterklassen:	
Protokoll:	
Auftrag	ISchlange
nachher	Die Schlange wurde initialisiert. Sie ist leer.
Auftrag	Einfuegen(neu: Basis)
nachher	Das Element „neu“ ist am Ende der Schlange angefügt.
Auftrag	Loeschen
vorher	Die Schlange ist nicht leer.
nachher	Das erste Element der Schlange ist gelöscht.
Anfrage	Element : Basis
vorher	Die Schlange ist nicht leer.
nachher	ZeigeElement ist das erste Element der Schlange.
Anfrage	Leer: Wahrheitswert
nachher	Leer ist wahr, wenn sich in der Schlange kein Element befindet.
Auftrag	GibFrei
nachher	Die Schlange steht nicht mehr zur Verfügung.

Musterlösung

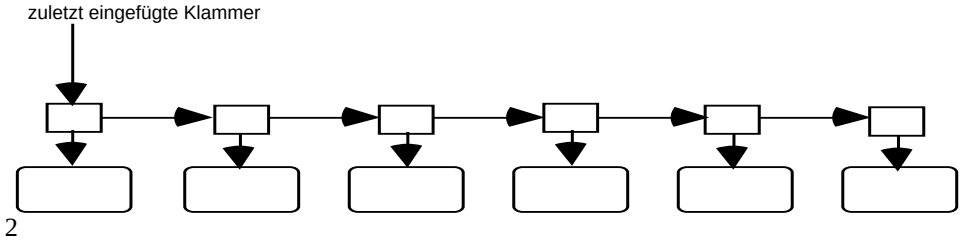
Zu a) Die Verkettungstechnik geht davon aus, dass jedes in den Stapel eingefügte Element seinen Vorgänger im Stapel kennt. In der folgenden Graphik ist das durch einen gerichteten Pfeil (Kennt-Beziehung) angedeutet.



1

Elemente, die in den Stapel eingefügt werden wollen, müssen also die Eigenschaft besitzen, ein weiteres Element kennen zu können. Wir haben solche Elemente Kettenelemente genannt. Ein Stapel, der so realisiert ist, kann also nur Kettenelemente aufnehmen. Ein Kettenelement, welches in einen bestimmten Stapel eingefügt wurde, kann nicht gleichzeitig auch in eine andere ähnliche Struktur, z. B. in einen anderen Stapel eingefügt werden.

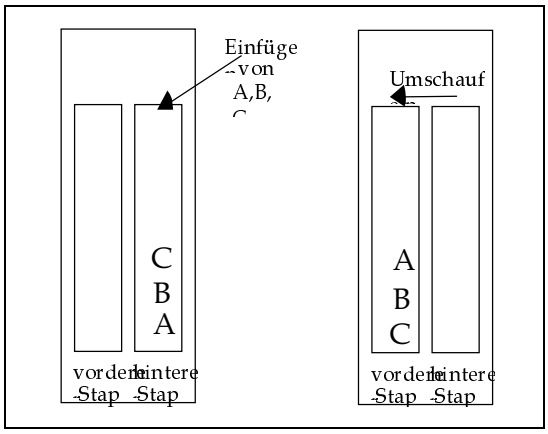
Im Gegensatz dazu kann man mit Hilfe der Klammertechnik jedes beliebige Element in einen Stapel einfügen. Zur Struktur des Stapels gehört eine Klammer, die die Verkettungsstruktur beinhaltet. Die eingefügten Elemente hängen an diesen Klammern, d.h. die Klammern kennen das geklammerte Element, das selbst keine weitere Struktur tragen muss. In der folgenden Graphik sind die Kennt-Beziehungen wie oben durch einen gerichteten Pfeil angedeutet.



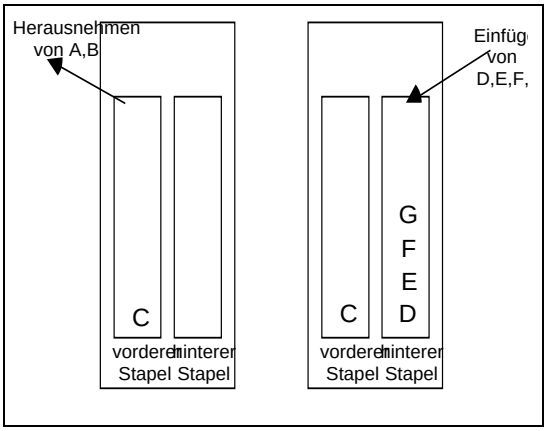
2

Ein Element, welches mit dieser Technik in einen Stapel eingefügt wurde, kann gleichzeitig Mitglied in beliebig vielen anderen Strukturen dieser Art sein.

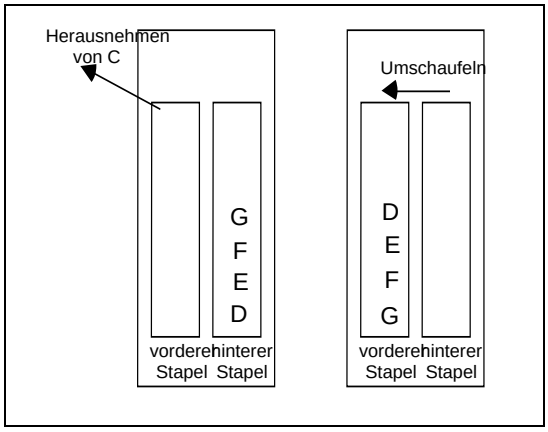
Zu b) 1.



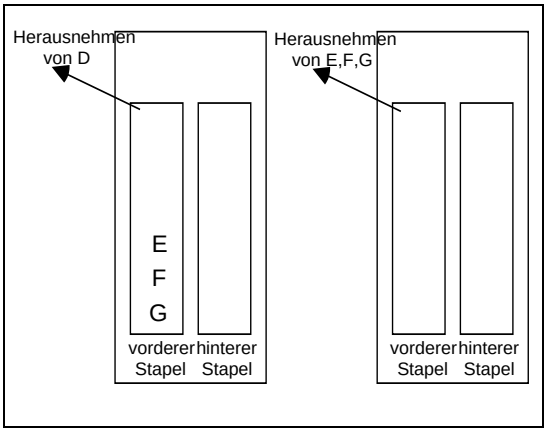
2.



3.



4.



Die Elemente werden bei diesem Modell in der Reihenfolge, in der sie eingefügt wurden, auch wieder herausgenommen. Das FIFO-Prinzip ist deutlich erkennbar. Offensichtlich muss immer dann umgeschaufelt werden, wenn beim Versuch ein Element herauszunehmen der vordere Stapel leer ist.

Zu c) Bei der Realisierung des in b) beschriebenen Modells besitzt eine Schlange zwei Stapel (Hat-Struktur). In Pascal ergibt sich folgende Realisierung:

```
kSchlange = object
   zVorderStapel,
   zHinterStapel:kStapel;
   procedure ISchlange;
   procedure Einfuegen(neu:kBasis);
   procedure Loeschen;
   function ZeigeElement:kBasis;
   function Leer:Boolean;
   procedure GibFrei;
   procedure Umschaufeln;
end;

procedure kSchlange.ISchlange;
begin
   New(zVorderStapel);
   zVorderStapel.IStapel;
   New(zHinterStapel);
   zHinterStapel.IStapel;
end;

procedure
    kSchlange.Einfuegen(neu:kBasis);
begin
   zHinterStapel.Einfuegen(neu)
end;

procedure kSchlange.Loeschen;
begin
   if zVorderStapel.Leer
      then SELF.Umschaufeln;
   zVorderStapel.Loeschen
end;

function
    kSchlange.ZeigeElement: kBasis;
begin
   ZeigeElement:=
      zVorderStapel.ZeigeElement
end;

function kSchlange.Leer:Boolean;
begin
   Leer:=
      zVorderStapel.Leer
      and
      zHinterStapel.Leer
end;

procedure kSchlange.GibFrei;
begin
   zVorderStapel.GibFrei;
   dispose(zVorderStapel);
   zHinterStapel.GibFrei;
   dispose(zHinterStapel)
end;

procedure kSchlange.Umschaufeln;
begin
   while not zHinterStapel.Leer do
   begin
      zVorderStapel.Einfuegen(
         zHinterStapel.ZeigeElement);
      zHinterStapel.Loeschen
   end;
end;
```

Aufgabe im schriftlichen Abitur (3. Beispiel, Themenbereich Datenbanken)

Lernvoraussetzungen und Anforderungen

Zu Teilaufgabe a):

Die aus dem Unterricht bekannten Definitionen der Normalformen müssen auf die für die Schülerinnen und Schüler neue Aufgabenstellung angewendet werden. Dazu wird ein Relationenschema für eine Teilmenge der Attribute vorgegeben. Für die Überprüfung dieses Schemas ist es notwendig, die aus der Anwendung entstehenden Abhängigkeiten zu erkennen. Entsprechend der Komplexität der Überlegungen sind aus unterschiedlichen Bewertungen von Abhängigkeiten unterschiedliche Aufteilungen des gegebenen Schemas möglich. In der Musterlösung ist ein möglicher Weg angegeben. Der Schwerpunkt bei der Lösung der Aufgabe ist also bei Kenntnissen aus dem Aufgabenbereich II zu sehen. Daneben sind jedoch Anforderungen aus den Bereichen I und auch III zu erkennen. Die Schülerinnen und Schüler können jedoch die weiteren Aufgabenteile lösen, ohne die volle Komplexität des Problems erkannt zu haben. Normalformen sind in Stufe 11/I eingeführt worden und wurden vertiefend in 13/I aufgegriffen.

Zu Aufgabe b):

Wie in Aufgabe a) sind auch hier verschiedene Lösungen denkbar. Die Schwierigkeit bei der Erstellung des ER-Modells liegt in der Festlegung der Entitys und der Relationen. So stellt z. B. die vorgegebene Tabelle Übungsleiter eine Relationentabelle dar, die eine Verbindung zwischen den Mitgliedern und den Übungsgruppen darstellt. Eine Bewertung als Entity führt zu einem anderen Diagramm. Außerdem ist festzulegen, wie die Verknüpfung zwischen den Sportstätten und den restlichen Informationen gestaltet werden kann. Es gelten hier dieselben Bemerkungen wie zur Teilaufgabe a). Obwohl die Erstellung von ER-Diagrammen im Unterricht sowohl in Stufe 11/I als auch vertiefend in Stufe 13/I behandelt wurde, ist eine schematische Übertragung von behandelten Problemen nicht möglich.

Zu Aufgabe c):

Im Unterricht wurden sowohl Abfragen in SQL wie in QbE behandelt. Bei der Abfrageformulierung in SQL wurde insbesondere der algorithmische Aspekt betrachtet. Neben Kenntnissen aus dem Anforderunsgbereich I bei der Verwendung der Schlüsselwörter sind zur Formulierung der Abfragen Kenntnisse aus dem Bereich II nötig. Insbesondere Teil 6 erfordert wegen der Notwendigkeit mehrere Tabellen zu verbinden, ein hohes Maß an Organisation. Die Kenntnisse stammen aus den Halbjahren 11/I und 13/I.

Zu Aufgabe d):

Indizierung und Verkettung von Daten wurde in der Stufe 12/I behandelt. Die Schülerinnen und Schüler müssen ihre Kenntnisse auf das ihnen unbekannte Beispiel anwenden. Die Aufgabenteile 1. und 2. sind also schwerpunktmäßig dem Aufgabenbereich II zuzuordnen. Dagegen soll in Teil 3. das Wissen über Vor- und Nachteile des Hash-Verfahrens auf eine in ähnlicher Form im Unterricht behandelte Problemstellung angewendet werden, sodass hier der Schwerpunkt im Bereich I liegt.

Zu Aufgabe e):

Die Schichten-Architektur wurde über den ganzen Kursverlauf von 11/I bis 13/II. immer wieder aufgegriffen. Deshalb dürfte es den Schülerinnen und Schülern nicht schwer fallen, diese Teilaufgabe zu behandeln. Diese Teilaufgabe ist den Bereichen I und II zuzuordnen.

Aufgabenstellung

Es wird eine Datenbank für einen Sportverein betrachtet. Der Verein hat mehrere Abteilungen, jedes Mitglied kann in mehreren Abteilungen (Sportarten) aktiv sein. Außerdem gibt es Übungsleiter, die bestimmte Trainingsgruppen leiten. Die Leiter sind Mitglieder des Vereins. Folgende Daten werden gespeichert:

Eine eindeutige Mitgliedsnummer, angebotene Sportarten, zu jeder Sportart Nummern der Übungsgruppen mit den Trainingszeiten, Mitgliedsnamen, -adresse, Bankverbindung (Kontonummer, Name der Bank, BLZ), Geschlecht, Geburtsdatum, Eintrittsdatum, Sportarten, in denen das Mitglied aktiv ist, für jede Sportart den zu zahlenden Beitrag, wobei der Beitrag für jede betriebene Sportart gezahlt werden, sowie eine Information, ob der Beitrag bereits gezahlt wurde, Übungsleiternamen mit ihrer Mitgliedsnummer, die Sportarten, in denen sie Übungsgruppen leiten, Trainingsort mit Adresse und Telefonnummer für jede Sportart.

a) Es wird als Erstes ein Relationenschema **Mitglied** (Name, Nummer, Adresse, Bankverbindung, Geschlecht, Gebdatum, Beitrag_gezahlt, Eintrittsdatum, Sportarten mit den Übungsgruppen, Übungszeiten) vorgeschlagen.
   1. Entscheiden Sie, ob das Schema der ersten Normalform genügt und verändern Sie es falls nötig!
   2. Geben Sie die funktionalen Abhängigkeiten an und überführen Sie das Schema in eine Form, die der zweiten und dritten Normalform genügt! Begründen Sie Ihre Veränderungen und die Festlegung der

Schlüssel! Zu welchen Schwierigkeiten kann es kommen, wenn die Normalisierungen nicht durchgeführt werden?

b) Als weitere Tabellen werden **Übungsleiter** (Mitgl.Nummer, Sportart, Übungsgruppennummer, Entschädigung) und **Trainingsort** (Name, Straße, Ort, Tel.Nummer, Hausmeister) vorgeschlagen. Erstellen Sie ein ER-Diagramm mit allen in a) betrachteten Entities und geben Sie die Wertigkeiten der Beziehungen an. Berücksichtigen Sie, soweit noch nicht geschehen, alle in der Einführung genannten Daten und ergänzen Sie, falls erforderlich noch Entities. Geben Sie eine vollständige Beschreibung aller Tabellen an.

c) Erstellen Sie die folgenden Abfragen in SQL:
   1. Geben Sie die Mitgliedsnummern aller Übungsleiter aus, die Übungsgruppen in den Sportarten Fußball oder Basketball leiten! Bei der Ausgabe sollen auch die Bezeichnungen der Übungsgruppe und die Sportarten angegeben werden.
   2. Geben Sie die Namen und Sportarten aller Übungsleiter aus, die vor dem 1.1.1980 geboren wurden.
   3. Der Mitgliedsbeitrag wird um 5% erhöht. Veranlassen Sie die Veränderung der Daten.
   4. Geben Sie für das Mitglied Aktivus die Höhe des insgesamt zu zahlenden Beitrages aus.
   5. Geben Sie die Namen aller Mitglieder mit den zugehörenden Bankverbindungen aus, die ihren Beitrag noch nicht gezahlt haben.
   6. Die Übungsgruppenleiterin Eifrig erleidet einen Bänderriss. Daher müssen die Mitglieder über den Ausfall der Übungsstunden informiert werden. Geben Sie eine Liste aller Übungsgruppen mit den Namen der Hallen und deren Telefonnummern aus, die Frau Eifrig leitet.

d) Die Übungsleitertabelle enthält folgende Daten:
   1. Die Daten sind nach dem Primärschlüssel sortiert. Zum schnellen Suchen der Sportart soll ein zusätzlicher Index mit alphabetisch sortierter Sportart eingerichtet werden. Erstellen Sie eine Index-Tabelle! Beschreiben Sie den Algorithmus zum Einfügen des neuen Datensatzes mit der Mitgliedsnummer 1500 und der Sportart Schwimmen!
   2. Die Verbindung zwischen den Übungsleitern und den Sportarten soll über Kettsätze aufgebaut werden. Skizzieren Sie diese Verbindung!
   3. Warum ist ein Zugriffsverfahren mit Hilfe der Hash-Methode für die Speicherung der Banken eine sinnvolle und für die Übungsleiter keine sinnvolle Möglichkeit?

Mitgliedsnummer	Sportart	Trainingsgruppe	Entschädigung
1156	Volleyball	1	145
1456	Basketball	1	65
1456	Volleyball	2	70
2876	Handball	1	115
2900	Basketball	2	250
2931	Schwimmen	1	135
2931	Gymnastik	1	115
2956	Badminton	1	50
3067	Fußball	1	235
3111	Fußball	2	1000

e) Geben Sie an, welche Schichten der Datenbank durch die obigen Aufgabenteile jeweils betroffen sind!

Musterlösung

zu Teilaufgabe a): Normalformen

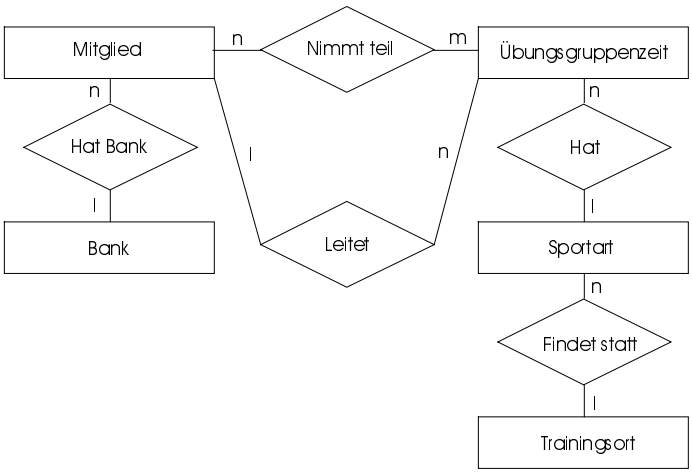
1. Beim Mitgliedsnamen wird sinnvollerweise Name und Vorname gespeichert. In der ersten Normalform muss diese Information in zwei Attribute aufgeteilt werden. Das Attribut Adresse erfüllt die Bedingungen der ersten Normalform nur dann, wenn es in Straße und Wohnort (evtl. mit integrierter PLZ) aufgeteilt wird. Dies gilt ebenso für die Bankverbindung: BLZ, Bankname, Kontonummer. Da ein Mitglied in mehreren Sportarten tätig sein kann, ist das Attribut Sportart mit den zugehörigen Übungsgruppen und Übungszeiten ein Aufzählungstyp, der gemäß erster Normalform nicht erlaubt ist. Es wird daher eine eigene Tabelle Mitgl_Sport erstellt, deren Attribute Mitgliedsnummer, Sportart, Übungsgruppe und Übungszeit sind. Gibt es zu einer Übungsgruppe mehrere Trainingszeiten, so diese Tabelle noch einmal aufzuteilen.

2. Folgende Attribute sind von der Mitgliedsnummer funktional abhängig: Mitgliedsnachname, Mitgliedsvorname, Straße, Wohnort, BLZ, Bankname, KontoNr., Geschlecht, Geb.Datum, Beitrag gezahlt, Eintrittsdatum,

Sportarten, Übungsgruppen, Übungszeiten. Der Bankname ist von der BLZ abhängig, die Übungszeiten hängen von Sportart und Übungsgruppe ab. Schlüssel für das betrachtete Schema ist also sinnvollerweise die Mitgliedsnummer, da es zwei Mitglieder mit demselben Namen geben kann. Da die Übungszeiten nur von Sportart und Übungsgruppe voll funktional abhängig sind, muss das in 1. entstandene Schema Mitgl_Sport unterteilt werden, damit es in zweiter Normalform ist: **Mitgl_Sport** (MitglNr, Sportart, Übungsgruppe) und **ÜbZeit** (Sportart, Übungsgruppe, Übungszeiten). Schlüssel des letzten Schemas sind Sportart und Übungszeit, da nur beide zusammen die Übungszeit bestimmen. Da der Bankname direkt von der Bankleitzahl abhängig ist, die BLZ jedoch kein Schlüssel in dem angegebenen Schema darstellt, muss zur Erfüllung der dritten Normalform diese Abhängigkeit durch Erstellung einer separaten Tabelle **Bank** (BLZ, Bankname) aus dem gegebenen Schema entfernt werden. Schlüssel ist die BLZ. Ohne die Veränderungen kann es zu Änderungsanomalien und Löschanomalien bei Bankdaten und bei den Übungszeiten kommen.

zu Teilaufgabe b): ER-Diagramm

Als Tabellen ergeben sich neben den bereits in a) beschriebenen Tabellen Mitglied, ÜbZeit, Bank als Entitys und Mitgl_Sport als Relation und den in der Aufgabenstellung angegebenen Tabellen Übungsleiter und Trainingsort noch die Tabelle zu der Relation „findet statt“ als Ort_Sportart mit den Attributen Trainingsort_Name und Sportart. Alle anderen Beziehungen können direkt über die Primärschlüssel aufgebaut werden.



zu Teilaufgabe c): Abfragen

```
1.Select MitglNr, Sportart, Übungsgruppe
  From Übungsleiter
  Where (Sportart = „Fußball“) OR (Sportart = „Basketball“);
2.Select Übungsleiter.MitglNr, Mitglied.Name, Mitglied.Vorname, Übungslei-
  ter.Sportart
  From Mitglied  Join Übungsleiter On Mitglied.MitglNr=Übungsleiter.MitglNr
  Where (MitglNr.GebDatum < „01.01.80“);
3.Update Sportart
  Set Beitragshöhe = Beitragshöhe * 1.05;
4.Select Mitglied.Name, Sum(Sportart.Beitragshöhe)
  From Mitglied  Join
      (Mitgl_Sport  Join Sportart On Mitgl_Sport.Sportart=Sportart.Sportart)
      ON Mitglied.MitglNr=Mitgl_Sport.MitglNr
  Group by Mitglied.Name
  Having (Mitglied.Name = „Aktivus“);
5.Select Mitglied.MitglNr, Mitglied.Name, Mitglied.Vorname, Bank.BLZ, Bank.Bankname
  From Bank Join Mitglied ON Bank.BLZ = Mitglied.BLZ
  Where (Mitglied.Beitrag_gezahlt = No);
6.Select Mitglied.Name, Übungsleiter.Sportart, Übungsleiter.Übungsgruppe, Trai-
  ningsort.Name,
      Trainingsort.TelNummer
  From Trainingsort Join
      (ÜbZeit Join
            (Mitglied Join
                  Übungsleiter ON Mitglied.MitglNr = Übungsleiter.MitglNr)
                        On (ÜbZeit.Übungsgruppe=Übungsleiter.Übungsgruppe) AND
                              (ÜbZeit.Sportart=Übungsleiter.Sportart)
```

```
    On Trainingsort.Sportart = ÜbZeit.Sportart
Where (Mitglied.name = „Eifrig“);
```

zu Teilaufgabe d): Datenzugriffe

1. Indizierung

In der Tabelle der Übungsleiter wird der neue Datensatz als vierter eingefügt. Dadurch ändern sich alle Indizes in der obigen Tabelle. Außerdem wird eine neue Zeile mit den Informationen „Schwimmen“ und dem Index „4“ in der obigen Tabelle eingefügt, indem die beiden letzten Zeilen aufgeschoben werden.

2. Kettsätze

Bei der Beziehung handelt es sich um eine m:n Beziehung, sodass die Verwendung von Kettsätzen sinnvoll ist. Es wird eine zusätzliche Liste mit zwei Zeigern aufgebaut, in der einerseits eine Verkettung über die Mitgliedsnummern und in einer zweiten Komponente eine Verkettung über die Sportarten gespeichert werden kann.

Sportart	Datensatz-Nr.
Badminton	2
Basketball	5
Basketball	8
Fußball	9
Fußball	10
Gymnastik	7
Handball	4
Schwimmen	6
Volleyball	1
Volleyball	3

Liste der Übungsleiter

Datensatz-Nr.	MitglNr	Listen-anfang
1	1156	1
2	1456	2
3	2876	4
4	2900	5
5	2931	6
6	1956	8
7	3067	9
8	3111	10

Liste der Sportarten

Datensatz-Nr	Sportarten	Listen-anfang
1	Badminton	8
2	Basketball	2
3	Fußball	9
4	Gymnastik	7
5	Handball	4
6	Schwimmen	6
7	Volleyball	1

In den Verkettungsfeldern geben die Zahlen die Verweise auf die Datensätze der Kettsätze an, m# beschreibt den Rückverweis auf die Übungsleiterliste und S# den Rückverweis auf die Liste der Sportarten.

MitglNr	Sportart	Verk.-Mitgl	Verk.-Sportart	Datensatz-Nr.
1156	Volleyball	m1	3	1
1456	Basketball	3	5	2
1456	Volleyball	m2	s7	3
2876	Handball	m3	s5	4
2900	Basketball	m4	s1	5
2931	Schwimmen	7	s6	6
2931	Gymnastik	m5	s4	7
2956	Badminton	m6	s1	8
3067	Fußball	m7	10	9
3111	Fußball	m8	s3	10

3. Hash-Verfahren

Da sich Bankleitzahl und Banknamen nur äußert selten ändern werden, wohl aber sicherlich neue Bankverbindungen hinzugefügt werden, ist das Hash-Verfahren ein geeignetes Verfahren. Änderungen bestehender Datensätze sind sehr aufwendig. Damit ist bei den Übungsleitern immer zu rechnen, sodass dafür dieses Speicherverfahren ungeeignet ist.

zu Teilaufgabe e): Schichten der Datenbank

Geht man von der Drei-Schichten-Architektur aus, so ist die Teilaufgabe d) der untersten Schicht dem internen Schema zuzuordnen. Teilaufgabe c) kann der externen Ebene zugeordnet werden, alle anderen Aufgabenteile ge-

hören zum konzeptionellen Schema, also zur zweiten Schicht. Dabei kann Teilaufgabe c) je nach Aufbau des DBMS auch der obersten Ebene zugeordnet werden.

5.4 Die mündliche Abiturprüfung

(1) Für die mündliche Prüfung gelten im Grundsatz die gleichen Anforderungen wie für die schriftliche Prüfung.
Die Prüfung ist insgesamt so anzulegen, dass die Prüflinge
- sicheres geordnetes Wissen
- Vertrautheit mit der Arbeitsweise des Faches
- Verständnis und Urteilsfähigkeit
- selbständiges Denken
- Sinn für Zusammenhänge des Fachbereichs
- Darstellungsvermögen

beweisen können.
Die Prüflinge sollen in einem ersten Teil selbständig die vorbereitete Aufgabe in zusammenhängendem Vortrag zu lösen versuchen. In einem zweiten Teil sollen vor allem größere fachliche und fachübergreifende Zusammenhänge in einem Prüfungsgespräch angesprochen werden.

(2) Die mündliche Prüfung bezieht sich - wie die schriftliche - auf den Unterricht der Qualifikationsphase. Sie darf sich nicht auf das Sachgebiet eines Kurshalbjahres beschränken. Sie muss also die Sachgebiete eines Kurshalbjahres überschreiten. Diese Forderung muss - falls ihr nicht bereits durch die für den 1. Prüfungsteil gestellte Aufgabe genügt wird - durch die Ausweitung des Prüfungsgesprächs auf größere fachliche Zusammenhänge im zweiten Prüfungsteil erfüllt werden.

(3) Die Prüfungsaufgaben müssen so angelegt sein, dass es den Prüflingen grundsätzlich möglich ist, jede Notenstufe zu erreichen.

(4) Für die gesamte Prüfung gilt:
- die zu prüfenden Sachbereiche dürfen nicht zwischen Prüferin oder Prüfer und den Prüflingen abgesprochen sein; Absprachen über Spezialgebiete sind unzulässig;
- eine Prüfung in Sachbereichen, die sich die Prüflinge außerhalb des Unterrichts privat erarbeitet haben, ist nicht zulässig;
- die mündliche Prüfung darf keine Wiederholung der schriftlichen Prüfung sein.

5.4.1 Aufgabenstellung für den ersten Teil der mündlichen Prüfung

(1) Für jede Prüfung ist den Prüflingen eine für sie neue, begrenzte Aufgabe zu stellen. Eine ausschließlich oder vorrangig auf Reproduktion ausgerichtete Aufgabe entspricht nicht den Prüfungsanforderungen.

(2) In der mündlichen Prüfung sollte weniger im Vordergrund stehen, ob die Prüflinge Begriffe und Methoden der Informatik anwenden können, als vielmehr ob sie sie zu verstehen gelernt haben. Bei der Themenwahl ist daher darauf zu achten, dass innerhalb größe-

rer fachlicher Zusammenhänge Begriffe, Algorithmen und Methoden vergleichend dargestellt, analysiert und kritisch beurteilt werden können.

(3) Die Aufgabenstellung sollte so angelegt sein, dass sich im zweiten Prüfungsteil ein Prüfungsgespräch über größere fachliche Zusammenhänge entwickeln kann.
Die Aufgabe wird den Prüflingen schriftlich vorgelegt. Es ist nicht zulässig, den Prüflingen gleichzeitig zwei oder mehrere voneinander abweichende Aufgaben zu stellen oder sie zwischen mehreren Aufgaben wählen zu lassen.

5.4.2 Aufgabenstellung für den zweiten Teil der mündlichen Prüfung

(1) Der zweite Teil der Prüfung besteht aus einem Prüfungsgespräch, das vor allem größere fachliche und fachübergreifende Zusammenhänge überprüfen soll.

(2) Der zweite Teil der Prüfung sollte etwa die Hälfte der Gesamtprüfungszeit in Anspruch nehmen.

5.4.3 Bewertung der Prüfungsleistungen

Für die Bewertung der Prüfungsleistungen gelten in der mündlichen Prüfung die gleichen Grundsätze wie für die schriftliche Prüfung. Dabei sind die der Prüfungsaufgabe zugrunde liegenden Anforderungsbereiche zu beachten (siehe Kapitel 5.2). Für das Prüfungsgespräch im zweiten Teil ergeben sich ergänzende Bewertungskriterien wie z. B.:

- richtiges Erfassen von Fachfragen,
- sach- und adressatengerechtes Antworten,
- Schwierigkeiten, die im Gespräch auftreten, erkennen und erläutern,
- weiterführende Fragestellungen, die sich im Verlauf des Gesprächs ergeben, sinnvoll einbringen und verarbeiten.

5.4.4 Beispiel für Prüfungsaufgaben in der mündlichen Prüfung

Thema: Rechtschreibhilfe/Vergleich von Datenstrukturen

Aufgabenstellung

Der Chef der Firma CeTe, Herr W. Olwo beauftragt Sie mit einer Vorabanalyse geeigneter Datenstrukturen und Algorithmen zu dem folgenden Problem:
Es soll über den Kern eines Programms nachgedacht werden, das Rechtschreibfehler erkennen kann: Für jedes Wort eines zu prüfenden Textes wird untersucht, ob dem Programm das Wort bekannt ist (d.h. ob es gespeichert ist) oder nicht. Ist es bekannt, dann enthält es auch keine Rechtschreibfehler und es wird weitergegangen. Ist es nicht bekannt, wird die Anwenderin bzw. der Anwender gefragt, ob das Wort einen Rechtschreibfehler enthält. Nur dann, wenn es keinen Rechtschreibfehler enthält, wird das Wort abgespeichert und ist dem Programm fortan bekannt. Das Programm „lernt“ also im Laufe der Verarbeitung neue Wörter hinzu. In die engere Auswahl kommen drei Datenstrukturtechniken:

(1) Geordnete lineare Listen
(2) Binäre Suchbäume
(3) Hashtabellen

Teilaufgaben:

a) Zu Beginn sei dem Programm noch kein Wort bekannt. Beschreiben Sie verbal und durch Skizze der Belegung der jeweiligen Datenstruktur, wie nacheinander die Wörter des Satzes „EINE ENTWICKLUNG VON DATENSTRUKTUREN UND ALGORITHMEN IST EINE KREATIVE TÄTIGKEIT VON INFORMATIKERN“
in die verschiedenen Datenstrukturen eingetragen werden. Erläutern Sie dabei Gemeinsamkeiten und Unterschiede der drei Datenstrukturen.

b) Schätzen Sie ab, mit welchem Aufwand eine Prozedur „teste und füge gegebenenfalls ein“ im besten, im schlechtesten und im durchschnittlichen Fall auf der Basis der drei Datenstrukturen bei einem maximalen Datenumfang von n jeweils abgearbeitet werden könnte. Bewerten Sie insgesamt die Eignung der drei Datenstrukturen zur Lösung des Teilproblems.

c) Die Daten sollen am Ende des Tages aus den Datenstrukturen in eine sequentielle Datei übertragen und am nächsten erneut eingelesen werden. Beschreiben Sie, wie der Ablauf in den drei Fällen konzipiert werden könnte und vergleichen Sie sie.

5.5 Die besondere Lernleistung

Die Absicht, eine besondere Lernleistung zu erbringen, muss spätestens am Ende der Jahrgangsstufe 12 bei der Schule bzw. bei der Schulleiterin oder beim Schulleiter angezeigt werden. Die Schulleitung entscheidet in Abstimmung mit der Lehrkraft, die als Korrektor vorgesehen ist, ob die beantragte Arbeit als besondere Lernleistung zugelassen werden kann. Die Arbeit ist nach den Maßstäben und dem Verfahren für die Abiturprüfung zu korrigieren und zu bewerten. In einem Kolloquium, das im Zusammenhang mit der Abiturprüfung nach Festlegung durch die Schule stattfindet, stellen die Prüflinge vor einem Fachprüfungsausschuss die Ergebnisse der besonderen Lernleistung dar, erläutern sie und antworten auf Fragen. Die Endnote wird aufgrund der insgesamt in der besonderen Lernleistung und im Kolloquium erbrachten Leistungen gewertet; eine Gewichtung findet nicht statt. Bei Arbeiten, an denen mehrere Schülerinnen und Schüler beteiligt waren, muss die individuelle Schülerleistung erkennbar und bewertbar sein.

Eine besondere Lernleistung im Fach Informatik kann die Entwicklung eines Softwarepaketes einschließlich aller notwendigen Dokumentationen sein, das sich in der Schule nutzbringend einsetzen lässt, z. B.

- ein Programm zur Verwaltung der Schulbibliothek
- ein Programm zur organisatorischen Unterstützung der Bundesjugendspiele
- Lernprogramme für verschiedene Fächer (Vokabeln, Rechenarten usw.)
- ein Compiler/Interpreter für einen eigenen Programmiersprachentwurf.

6. Hinweise zur Arbeit mit dem Lehrplan

Aufgaben der Fachkonferenzen

Nach § 7 Abs. 3 Nr. 1 des Schulmitwirkungsgesetzes entscheidet die Fachkonferenz über
- Grundsätze zur fachmethodischen und fachdidaktischen Arbeit sowie über
- Grundsätze zur Leistungsbewertung.

Die Beschlüsse der Fachkonferenz gehen von den im vorstehenden Lehrplan festgelegten obligatorischen Regelungen aus und sollen die Vergleichbarkeit der Anforderungen sicherstellen. Hierbei ist zu beachten, dass die Freiheit und Verantwortung der Lehrerinnen und Lehrer bei der Gestaltung des Unterrichts und der Erziehung durch Konferenzbeschlüsse nicht unzumutbar eingeschränkt werden dürfen (§ 3 Abs. 2 SchMG).

Die Fachkonferenz berät und entscheidet in den im Folgenden genannten Bereichen:
- Präzisierung der fachlichen Obligatorik und Maßnahmen zur Sicherung der Grundlagenkenntnisse
- Absprachen zu den fachspezifischen Grundlagen der Jahrgangsstufe 11
- Absprachen über die konkreten fachspezifischen Methoden und die konkreten Formen selbständigen Arbeitens
- Absprachen über den Rahmen von Unterrichtssequenzen
- Absprachen über die Formen fachübergreifenden Arbeitens und den Beitrag des Faches zu fächerverbindendem Unterricht
- Koordination des Einsatzes von Facharbeiten
- Absprachen zur „besonderen Lernleistung“.

Grundsätze zur Leistungsbewertung

Grundsätze und Formen der Lernerfolgsüberprüfung sind in Kapitel 4 behandelt worden. Es ist die Aufgabe der Fachkonferenz diese Grundsätze nach einheitlichen Kriterien umzusetzen.

Beschlüsse beziehen sich
- auf den breiten Einsatz von Aufgabentypen
- auf das Offenlegen und die Diskussion der Bewertungsmaßstäbe
- auf gemeinsam gestellte Klausurthemen/Abituraufgaben
- auf die beispielhafte Besprechung korrigierter Arbeiten.

Beiträge der Fachkonferenzen zur Schulprogrammentwicklung und zur Evaluation schulischer Arbeit

Aussagen zum fachbezogenen und fachübergreifenden Unterricht sind Bestandteil des Schulprogramms. Die Evaluation schulischer Arbeit bezieht sich zentral auf den Unterricht und seine Ergebnisse. Die Fachkonferenz spielt deshalb eine wichtige Rolle in der Schulprogrammarbeit und bei der Evaluation des Unterrichts. Dabei sind Prozesse und Ergebnisse des Unterrichts zu berücksichtigen. Die Fachkonferenz definiert die Evaluationsaufgaben, gibt Hinweise zur Lösung und leistet insoweit ihren Beitrag zur schulinternen Evaluation.