RheinlandPfalz

Lehrplanentwurf Informatik

Grund- und Leistungsfach
Einführungsphase und Qualifikationsphase
der gymnasialen Oberstufe
(Mainzer Studienstufe)

Ministerium für Bildung, Wissenschaft, Jugend und Kultur

1 Fachdidaktische Konzeption

1.1 Beitrag des Fachs Informatik zur Bildung

Informatik – ein Schulfach mit einer Schlüsselrolle

Informations- und Kommunikationstechnologien sind ein wesentlicher Bestandteil unseres gesellschaftlichen Lebens geworden. Sie ermöglichen es, Informationen zu jeder Zeit an praktisch jedem Ort verfügbar zu machen, riesige Datenmengen automatisiert zu verarbeiten und Nachrichten mit rasanter Geschwindigkeit zu übermitteln. Mit der fortschreitenden Ausbreitung drängen diese Technologien in fast alle Bereiche des gesellschaftlichen und privaten Lebens und betreffen somit inzwischen jeden. Der Umgang mit digital dargestellter Information und die Beherrschung von Informations- und Kommunikationssystemen werden infolgedessen als unverzichtbare Ergänzung der traditionellen Kulturtechniken Lesen, Schreiben und Rechnen angesehen. Die hierzu erforderlichen Kompetenzen werden heute ebenso wie Lese-, Schreib- und Rechenkompetenzen vielfach zur Gestaltung des gesellschaftlichen und privaten Lebens benötigt. Zu den Aufgaben einer allgemein bildenden Schule muss es daher gehören, diese Kompetenzen gezielt zu entwickeln, um Schülerinnen und Schülern die Orientierung in einer technisierten Welt und den Zugang zu allen Bereichen einer Informations- und Wissensgesellschaft zu ermöglichen. Alle Fächer können hierzu Beiträge leisten, indem sie Informations- und Kommunikationssysteme zur Bearbeitung spezifischer Aufgaben einsetzen. Den entscheidenden Beitrag zur Ausbildung entsprechender Kompetenzen, die auf einem vertieften Verständnis automatisierter Informationsverarbeitung beruhen und über bloße Bedienerfertigkeiten hinausgehen, kann aber nur der Informatikunterricht leisten. In diesem Sinne übernimmt der Informatikunterricht eine Schlüsselrolle innerhalb der Schule. Im Folgenden soll der spezifische Beitrag, den das Fach Informatik zur Bildung von Schülerinnen und Schülern leistet, weiter ausdifferenziert werden.

Informatik – ein Schulfach mit eigenem Profil

Informatik versteht sich als Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen, besonders der automatischen Verarbeitung mithilfe von Computern[1]. Informatik stellt demnach den fachlichen Kontext bereit, der für einen verständigen Umgang mit automatisierter Informationsverarbeitung und der hierzu benötigten Infrastruktur benötigt wird. Im Folgenden soll der Begriff „Informatiksystem“ benutzt werden, um Systeme zu bezeichnen, die Hard- und Software sowie die gesamte Infrastruktur zum Austausch von Daten umfassen können.

Der Beitrag des Informatikunterrichts zur Bildung von Schülerinnen und Schülern lässt sich unter folgenden Perspektiven beschreiben[2]:

- **Interaktion mit Informatiksystemen**
  Im Informatikunterricht lernen Schülerinnen und Schüler handelnd den flexiblen Umgang mit digital dargestellter Information, indem sie mit jeweils geeigneten Informatiksystemen interagieren. Sie nutzen diese Systeme als Werkzeuge, um sich Informationen zu beschaffen und zu verwalten und um Informationen in vielfältiger Weise zu bearbeiten.

[1] vgl. Duden Informatik 2003

[2] vgl. Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen, erarbeitet vom Fachausschuss 7.3 "Informatische Bildung in Schulen" der Gesellschaft für Informatik e.V.

- **Wirkprinzipien von Informatiksystemen**
  Im Informatikunterricht entwickeln Schülerinnen und Schüler ein Grundverständnis über die Wirkprinzipien von Informatiksystemen. Sie erlernen hierzu Konzepte und Modelle der Informatik, mit deren Hilfe die Systemzusammenhänge und Funktionsprinzipien der Komponenten erklärt und verstanden werden können. Dieses Grundverständnis erst ermöglicht eine über Bedienerfertigkeiten hinausgehende verständige Interaktion mit Informatiksystemen.
- **Modellierung von Informatiksystemen**
  Im Informatikunterricht erfahren Schülerinnen und Schüler, wie Informatiksysteme entworfen und realisiert werden können. In überschaubaren Problembereichen erfahren sie die Bedeutung informatischer Modellierung und die Schwierigkeiten bei der Realisierung zuverlässig funktionierender Systeme.
- **Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft**
  Im Informatikunterricht reflektieren und bewerten Schülerinnen und Schüler die Wechselwirkungen zwischen Informatiksystemen, den mit diesen interagierenden Individuen und der hiervon betroffenen Gesellschaft. Sie beschäftigen sich mit menschengerechter Technikgestaltung sowie ethischen und rechtlichen Fragen. Der Informatikunterricht ermöglicht eine fachlich fundierte Auseinandersetzung mit wichtigen gesellschaftlichen Aspekten und trägt so zu einer kompetenten und verantwortungsbewussten Nutzung der Informationstechnik bei.

Informatik – ein Schulfach mit allgemeinbildendem Anspruch

Der Informatikunterricht übernimmt nicht nur fachspezifische Aufgaben, indem er Schülerinnen und Schülern einen verständigen Umgang mit automatisierter Informationsverarbeitung und der hierzu benötigten Infrastruktur ermöglicht. Der Informatikunterricht leistet auch einen entscheidenden Beitrag zur Allgemeinbildung, indem er universell einsetzbare und längerfristig relevante Kompetenzen fördert, die in einer komplexer werdenden Welt zum Bewältigen von Problemen benötigt werden.

Ein zentraler Problembereich betrifft das Verstehen und Beherrschen komplexer Systeme. Gerade hier kann der Informatikunterricht entscheidend zur Entwicklung fächerübergreifender Kompetenzen und allgemeiner Problemlösestrategien beitragen.

Im Informatikunterricht lernen Schülerinnen und Schüler, komplexere Systeme zu strukturieren, sie insbesondere in überschaubare Teilsysteme zu zerlegen, um damit das Gesamtsystem durchschaubar zu machen.

Des Weiteren lernen sie, reale Systeme abstrahierend durch Modelle zu beschreiben, um sie einer weiteren Bearbeitung zugänglich zu machen. Dabei kommen Techniken zum Tragen, die auch in vielen anderen Wissenschaftsbereichen benutzt werden, wie z. B. Techniken zur funktionalen Modellierung von Abhängigkeiten, zur logischen Beschreibung von Zusammenhängen oder zur strukturellen Organisation von Abläufen.

Die Realisierung funktionierender Systeme erfordert sehr viel Sorgfalt und Genauigkeit in der Detailarbeit und oft auch ein beträchtliches Maß an Ausdauer. Hierbei erfahren Schülerinnen und Schüler auch, dass selbst entwickelte Systeme systematisch überprüft und kritisch beurteilt werden müssen, bevor sie an weitere Nutzer übergeben werden können.

Größere Systeme können nicht mehr von einzelnen Personen erstellt werden. Nur im Team lassen sich größere Aufgaben arbeitsteilig in einem vorgegebenen Zeitrahmen erledigen. Indem

Schülerinnen und Schüler im Informatikunterricht solche Aufgaben im Team lösen, werden Fähigkeiten zur Interaktion und Kommunikation mit anderen gefördert. Zudem werden wichtige Eigenschaften wie Zuverlässigkeit und Verantwortungsbereitschaft gefördert, die im späteren Berufsleben unerlässlich sind.

Informatik – ein Schulfach, das sich etabliert

Das Schulfach Informatik ist auf dem Weg, sich im Fächerkanon der Schule als gleichwertig zu anderen Fächern zu etablieren. Nur dadurch kann der Anspruch der Gesellschaft eingelöst werden, dass jede Schülerin und jeder Schüler während der Schulzeit die Möglichkeit erhält, informatische Bildung zu erwerben.

Das Schulfach Informatik kann derzeit – je nach Angebot – in der Sekundarstufe I im Rahmen des Wahlfachs / Wahlpflichtfachs und in der Sekundarstufe II im Rahmen eines Grund- oder Leistungsfachs belegt werden. Da Informatik noch kein Pflichtfach in der Sekundarstufe I ist, ergeben sich besondere Abhängigkeiten, die im Folgenden kurz erläutert werden sollen.

Das Wahlfach / Wahlpflichtfach ist so konzipiert, dass Schülerinnen und Schüler hier einerseits informatische Grundkompetenzen erwerben, andererseits aber auch ein erstes Bild der Informatik gewinnen und somit Grundlagen für eine Vertiefung im Leistungsfach gelegt werden. Das Wahlfach / Wahlpflichtfach geht davon aus, dass spezifische Fähigkeiten und Fertigkeiten, wie sie etwa bei der Benutzung von Anwendungsprogrammen benötigt werden, bereits im Vorfeld erworben worden sind.

In der Sekundarstufe II kann das Fach Informatik als Grund- oder Leistungsfach gewählt werden.

Informatik als Leistungsfach können nur diejenigen Schülerinnen und Schüler wählen, die in den vorangegangen zwei Klassenstufen am Wahlfach / Wahlpflichtfach Informatik teilgenommen haben. Über Ausnahmen in Einzelfällen entscheidet die Schulleitung auf Vorschlag der Fachschaft Informatik. Dabei muss sichergestellt sein, dass der Kenntnisstand der betreffenden Schülerin / des betreffenden Schülers einen erfolgreichen Besuch des Leistungsfaches ermöglicht.

Informatik als Grundfach setzt keine Kenntnisse aus dem Wahlfach / Wahlpflichtfach der Sekundarstufe I voraus, wohl aber Fähigkeiten und Fertigkeiten im Umgang mit dem Computer und mit Anwendungsprogrammen. In Schulen, in denen das Wahlfach / Wahlpflichtfach angeboten wird, können somit Schülerinnen und Schüler mit unterschiedlichen Voraussetzungen das Grundfach belegen. Wie diese systembedingte Heterogenität zu handhaben ist, wird unten genauer erläutert.

1.2 Inhaltliche Konzeption des Lehrplans

Der Lehrplan für das Grund- und Leistungsfach Informatik konkretisiert die Kompetenzen, die Schülerinnen und Schüler zur Nutzung, Erklärung, Entwicklung und Beurteilung von Informatiksystemen benötigen und im Informatikunterricht der Sekundarstufe II erwerben sollen. Zudem legt er die Inhalte fest, mit denen Schülerinnen und Schüler sich im Informatikunterricht auseinandersetzen sollen.

Inhaltsbereiche

Kompetenzen und Inhalte werden nach fachsystematischen Gesichtspunkten zusammengestellt. Die Gruppierung in Inhaltsbereiche soll nur ihre Darstellung im Lehrplan erleichtern. Sie ist keinesfalls als Beschreibung einer Abfolge von Unterrichtseinheiten gedacht. So wird durch die Anordnung in der Darstellung weder eine Zuordnung zu Schulhalbjahren beabsichtigt noch eine Reihenfolge der Bearbeitung nahe gelegt. Eine Beschäftigung mit den aufgeführten Inhalten sollte in thematisch ausgerichteten Unterrichtsreihen erfolgen, wobei insbesondere Möglichkeiten der Vernetzung genutzt werden sollen (s. u.). Folgende Inhaltsbereiche sind vorgesehen:

Grundfach	Leistungsfach
➢ Information und ihre Darstellung	➢ Sprachen und Automaten
➢ Aufbau und Funktionsweise eines Rechners	➢ Aufbau und Funktionsweise eines Rechners
➢ Kommunikation in Rechnernetzen	➢ Kommunikation in Rechnernetzen
➢ Algorithmisches Problemlösen	➢ Algorithmen und Datenstrukturen
	➢ Grenzen algorithmisch arbeitender Systeme
➢ Informatische Modellierung	➢ Informatische Modellierung
	➢ Deklarative Programmierung
➢ Software-Entwicklung	➢ Software-Entwicklung
	➢ Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft

Die teils differierenden inhaltlichen Akzentsetzungen im Grund- und Leistungsfach erklären sich durch die unterschiedlichen Voraussetzungen und die unterschiedlichen Zielsetzungen (s. u.).

Die Inhaltsbereiche sind so konzipiert, dass eine vertiefende Auseinandersetzung mit den hier aufgelisteten Inhalten zur intendierten informatischen Bildung führt, wie sie durch die bereits genannten Perspektiven „Interaktion mit Informatiksystemen“, „Wirkprinzipien von Informatik-

systemen“, „Modellierung von Informatiksystemen“, „Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft“ umschrieben werden.

Inhaltsbereiche und informatische Bildung

Interaktion mit Informatiksystemen spielt bei der unterrichtlichen Umsetzung aller Inhaltsbereiche eine wichtige Rolle. Bei der Erarbeitung und Vertiefung der inhaltlichen Zusammenhänge kommt eine Vielzahl von Software-Werkzeugen zum Einsatz, zum Beispiel: Mächtige Editoren erleichtern die Darstellung von Information in Texten und Bildern, geeignete Simulationsprogramme machen den Aufbau eines Rechners bzw. Funktionselemente von Kommunikationssystemen durchsichtiger, Experimente mit vorgegebenen und mit selbst entwickelten Programmen machen abstrakte Konzepte konkret erfahrbar, moderne Programmiersprachen mit integrierten Entwicklungsumgebungen unterstützen in mannigfacher Weise die Entwicklung, Implementierung und Validierung von algorithmischen Problemlösungen.

Aus der Interaktion mit Informatiksystemen erwächst vielfach der Wunsch, die Wirkprinzipien dieser Systeme zu verstehen. Umgekehrt ermöglicht erst das Verständnis dieser Prinzipien eine verständige Interaktion. Die Inhaltsbereiche zu Aufbau und Funktionsweise eines Rechners und zur Kommunikation in Rechnernetzen widmen sich in besonderer Weise diesen Wirkprinzipien. Aber auch in anderen Inhaltsbereichen kommen wichtige Prinzipien zum Tragen – so etwa bei der Codierung von Information oder der algorithmischen Verarbeitung der so gewonnenen Daten. Im Leistungsfach werden zudem weiterführende Fragen zur Tragweite informatischer Konzepte geklärt, etwa zur Verarbeitbarkeit formaler Sprachen mit Automaten.

Modellierung hat in der Informatik einen bedeutenden Stellenwert. Diese Tatsache spiegelt sich im Lehrplan durch einen eigenständigen Inhaltsbereich wieder. Modelle spielen aber auch in anderen Inhaltsbereichen eine wichtige Rolle. So werden etwa endliche Automaten durchgängig als Modelle benutzt, um Systemverhalten adäquat zu beschreiben.

Während der Komplex „Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft“ im Leistungsfach mit einem eigenen Inhaltsbereich ausgewiesen ist, werden diese Aspekte im Grundfach in andere Inhaltsbereiche integriert: Der Umgang mit Informationen wirft in natürlicher Weise Fragen des Datenschutzes und des rechtlich einwandfreien Umgangs mit Daten auf. Besonders das Internet beeinflusst den Einzelnen und die Gesellschaft so stark, dass neben ethischen und rechtlichen auch Sicherheitsfragen erörtert werden müssen. Unzuverlässige Software kann großen individuellen und gesellschaftlichen Schaden anrichten. Umgekehrt liegt in guter Software ein großes Potenzial zur Verbesserung der Lebensqualität. Im Umgang mit fertiger oder auch selbst erstellter Software lernen Schülerinnen und Schüler Chancen und Risiken der Informationstechnik fachlich fundiert einzuschätzen.

1.3 Darstellung der Inhaltsbereiche

Kompetenzen und Inhalte

Der Lehrplan konkretisiert aufbauend auf die Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA) das Abschlussprofil der Sekundarstufe II im Fach Informatik. Hierzu wird in jedem Inhaltsbereich eine inhaltsbezogene Kompetenz einschließlich der durch sie umfassten Teilkompetenzen beschrieben. Die Schülerinnen und Schüler sollen am Ende des Informatikunterrichts über alle in den Inhaltsbereichen aufgeführten Kompetenzen verfügen.

Im Grundfachlehrplan findet man beispielsweise innerhalb des Inhaltsbereichs „Information und ihre Darstellung“ die folgende Kompetenzbeschreibung:

Information zur Weiterverarbeitung in Informatiksystemen aufbereiten und sachgerecht Information aus den Verarbeitungsergebnissen gewinnen

Dazu gehört:

- *Information adäquat zur Weiterverarbeitung mit dem Computer darstellen*
- *Binäre Darstellung von Daten erläutern*
- *Rechtliche Aspekte beim Umgang mit Information beachten*
- *Datenbanken zur Informationsgewinnung nutzen*
- *Datenerhebungen unter dem Aspekt Datenschutz bewerten*

Kompetenzen können nur in Auseinandersetzung mit geeigneten Inhalten erworben werden. Zu jeder Teilkompetenz führt der Lehrplan verbindliche Inhalte auf, die im Unterricht bearbeitet werden müssen. Diese verbindlich vorgegebenen Inhalte sind bewusst allgemein formuliert und bieten daher Freiräume bei der Gestaltung des Unterrichts.

Zusätzlich zum verbindlichen inhaltlichen Rahmen gibt der Lehrplan konkrete Hinweise für eine mögliche unterrichtliche Umsetzung. Sie sollen zum einen Hilfen bei der didaktischen und methodischen Ausgestaltung des Unterrichts anbieten, zum anderen die Intensität der Beschäftigung mit den Inhalten implizit beschreiben. Die Hinweise haben orientierenden Charakter und skizzieren nur einen von mehreren möglichen Unterrichtsgängen.

Zur Verdeutlichung soll folgender Ausschnitt aus dem Inhaltsbereich „Information und ihre Darstellung“ im Grundfachlehrplan etwas näher erläutert werden.

Darstellung mit formalen Sprachen	➢ Mit Hilfe geeigneter Validierer die syntaktische Korrektheit von Webseiten überprüfen lassen und dabei herausarbeiten, dass die zur Erstellung von Webseiten benutzte Auszeichnungssprache (XHTML) eine formale Sprache ist, bei der strenge Syntaxregeln beachtet werden müssen. ➢ Zur weiteren Vertiefung einen Einblick in die Festlegung von Auszeichnungssprachen mit Hilfe von Dokumententypdefinitionen in XML gewinnen als Möglichkeit zur exakten Definition der Syntax einer Auszeichnungssprache erkunden. ➢ Syntax und Semantik als wesentliche Aspekte einer (formalen) Sprache herausstellen und in verschiedenen Kontexten (wie Informationsdarstellung mit Webseiten; Erstellung von Programmen) thematisieren.

Verbindlicher Inhalt ist hier die Darstellung von Information mit formalen Sprachen. Diese allgemeine Formulierung lässt sehr viele unterrichtliche Konkretisierungen zu. Die Hinweise in der rechten Spalte beschreiben eine mögliche Konkretisierung bei der Erstellung von Webseiten.

Teilaspekte des verbindlichen Inhalts „Darstellung mit formalen Sprachen“ sind etwa Syntax und Semantik sowie Darstellung und Überprüfung von Syntaxregeln. Eine mögliche Art der Auseinandersetzung besteht darin, syntaktische Aspekte experimentell mit Hilfe von Validierern zu erschließen und exemplarisch anhand von Dokumententypdefinitionen im Rahmen von XML zu vertiefen. Mit der Formulierung „Einen Einblick in ... gewinnen“ soll darauf hingewiesen werden, dass keine ausufernde Beschäftigung mit XML intendiert ist. Aus dem Umfang der Hinweise geht hier hervor, dass das Thema „Darstellung mit formalen Sprachen“ vertiefter behandelt werden soll.

1.4 Vernetzung der Inhaltsbereiche

Kompetenzen und die zugehörigen Inhalte können in verschiedener Reihenfolge im Unterricht entwickelt bzw. behandelt werden. Für den Unterricht soll sich die Auswahl der Lerninhalte in der Regel an der Bearbeitung konkreter Probleme orientieren. Dabei sollen Möglichkeiten der Vernetzung verschiedener Inhaltsbereiche genutzt werden. Zwei Beispiele sollen dies hier verdeutlichen.

Um etwa die Arbeitsweise eines Validierers zu verstehen, bietet es sich an, zu einem späteren Zeitpunkt in der Programmierphase einen einfachen Validierer selbst zu entwickeln. Zum Beispiel könnte ein E-Mail-Adress-Validierer zustandsbasiert modelliert und anschließend implementiert werden. Hierdurch werden verschiedene Inhaltsbereiche wie „Informatische Modellierung“ und „Information und ihre Darstellung“ miteinander vernetzt.

Um zu durchschauen, wie die Sicherheit von elektronischen Kommunikationsvorgängen heute gewährleistet wird, ist eine Beschäftigung mit kryptografischen Verfahren erforderlich. Dabei wird die zentrale Bedeutung des Sicherheitsaspekts für den Einzelnen und die gesamte Gesellschaft diskutiert. Zur Verdeutlichung der Vorgänge beim Chiffrieren und Dechiffrieren kann ein einfaches Chiffriersystem objektorientiert modelliert und implementiert werden. Dabei ist es bei modernen Verfahren erforderlich, sich mit Algorithmen zum schnellen Potenzieren zu beschäftigen.

Eine solche vernetzende Behandlung verdeutlicht verschiedene Facetten eines informatischen Themas und beugt einem Denken „in Schubladen“ vor.

Eine Vernetzung von Inhaltsbereichen ist allerdings nicht bei jedem Thema möglich bzw. in jeder Unterrichtseinheit sinnvoll. So ist es zur Vertiefung einzelner Themen im Unterricht oft erforderlich, bestimmte Inhalte fokussiert und isoliert zu bearbeiten.

Zeitansätze

Der Lehrplan weist keine Zeitansätze zur Bearbeitung von Inhaltsbereichen aus. Je nach Problemstellung können durch unterschiedliche Akzentuierungen bzw. durch Vernetzung von Inhaltsbereichen die Zeitwerte sehr stark variieren.

Die Lehrerin bzw. der Lehrer muss in eigener Verantwortung über die Zeitaufteilung entscheiden und dafür Sorge tragen, dass in der gesamten zur Verfügung stehenden Zeit alle aufgeführten Kompetenzen erworben werden können.

Freiräume / Addita

Die Darstellung im Grundfach weist neben verbindlichen Inhalten auch sogenannte Addita auf. Diese Addita können zur Gestaltung der Freiräume genutzt werden, insbesondere dann, wenn verbindliche Inhalte im Grundfach verkürzt behandelt werden. Letzteres kann nur dann zum Tragen kommen, wenn die Schule das Wahlfach Informatik in der Sekundarstufe I anbietet und Schülerinnen und Schüler, die das Wahlfach besucht haben, hier bereits umfassendere Grundkompetenzen erworben haben. Je nach Zusammensetzung und Vorkenntnissen der Lerngruppe können dann verbindliche Inhalte im Unterricht in reduziertem Umfang behandelt werden. Allerdings muss sichergestellt werden, dass Schülerinnen und Schüler, die das Wahlfach nicht besucht haben, hinreichend Gelegenheit erhalten, sich angemessen mit diesen Inhalten zu beschäftigen.

1.5 Differenzierung zwischen Grund- und Leistungsfach

Während im Grundfach Informatik die „Vermittlung einer wissenschaftspropädeutisch orientierten Grundbildung“ im Vordergrund steht, soll im Leistungsfach die „systematische, vertiefte und reflektierte wissenschaftspropädeutische Arbeit“ den Unterricht prägen.[3] Diese Forderung spiegelt sich im Lehrplan in vielfacher Weise wider.

Grund- und Leistungsfach Informatik gehen von unterschiedlichen Vorkenntnissen der Schülerinnen und Schüler aus: Im Grundfach wird – anders als im Leistungsfach – der vorherige Besuch des Wahlfachs / Wahlpflichtfachs nicht vorausgesetzt. Schülerinnen und Schüler, die das Grundfach belegen, müssen in aller Regel somit erst die grundlegenden Fragestellungen, Gegenstände und Arbeitsmethoden des neuen Fachs kennen lernen, während Schülerinnen und Schüler im Leistungsfach auf den im Wahlfach / Wahlpflichtfach erworbenen Einsichten und Kompetenzen aufbauen können. Besonders deutlich zeigt sich dies im Lehrplan dadurch, dass der im Grundfach zu behandelnde Inhaltsbereich „Information und ihre Darstellung“ im Leistungsfach nicht explizit aufgeführt ist, da wesentliche Elemente dieses Inhaltsbereichs bereits Gegenstand im Wahlfach / Wahlpflichtfach sind.

Die in der Sekundarstufe I geschaffenen Voraussetzungen und der insgesamt zur Verfügung stehende größere Zeitrahmen ermöglichen es, im Leistungsfach zusätzliche inhaltliche Schwerpunkte zu setzen. Beispielsweise ist im Leistungsfach die Behandlung des Inhaltbereichs „Deklarative Programmierung“ verpflichtend, während dieser gesamte Inhaltsbereich für das Grundfach entfällt. Auch innerhalb einzelner Inhaltsbereiche zeigen sich quantitative Unterschiede. Im Leistungsfach wird etwa – im Gegensatz zum Grundfach – Rekursion als algorithmische Grundstruktur verbindlich vorgegeben.

Wichtiger als die quantitativen sind die qualitativen Unterschiede, die sich in der Art und Weise zeigen, wie im Grund- bzw. Leistungsfach informatische Fragestellungen bearbeitet werden.

Die Vorgabe einer „systematischen, vertieften und reflektierten wissenschaftspropädeutischen Arbeit“ im Leistungsfach spiegelt sich dabei wie folgt im Lehrplan wider:

- Themen wie „formale Sprachen“ oder „Grenzen algorithmischer Verfahren“ werden im Leistungsfach systematischer betrachtet, während im Grundfach eine exemplarische Betrachtung ausreicht.
- Die systematische Behandlung bedingt oft eine abstraktere und auch formalere Beschreibung von Sachverhalten. So werden etwa abstrakte Verarbeitungsmodelle und hierauf aufbauend exakt definierte Begriffe im Leistungsfach eingeführt, um die Grenzen algorithmischer Verfahren zu verdeutlichen.
- Bei der Bearbeitung von Problemen wird im Leistungsfach ein höherer Komplexitätsgrad angestrebt als im Grundfach. Dies zeigt sich an z. B. an der Formulierung der Kompetenz im Inhaltsbereich „Informatische Modellierung“: Während die Schülerinnen und Schüler im Leistungsfach „zustandsbasierte Modelle zu *komplexeren* Problembereichen entwickeln“ können sollen, wird im Grundfach erwartet, dass sie „zustandsbasierte Modelle zu *einfachen* Problembereichen entwickeln“ können.

[3] vgl. EPA Informatik i.d.F. vom 5.2.2004, S. 6

- Die Erwartungen an die Selbstständigkeit bei der Problembearbeitung sind im Leistungsfach höher als im Grundfach, wie sich etwa im Inhaltsbereich „Software-Entwicklung“ bei der Gestaltung von Projekten zeigt.

Der Grad der Vertiefung ergibt sich oft erst implizit durch die Beschreibung einer möglichen Umsetzung. Auch wenn Kompetenzen bzw. verbindliche Inhalte im Grund- und Leistungsfachlehrplan gleich formuliert sind, ergeben sich hieraus nicht zwangsläufig gleiche Erwartungen an die Schülerinnen und Schüler. So wird z. B. nur im Leistungsfachlehrplan in den Hinweisen zur unterrichtlichen Umsetzung zum Inhalt „Moderne Verfahren zur Verschlüsselung und Signierung“ vorgeschlagen, das Grundprinzip des RSA-Verfahrens zu erarbeiten.

Neben allen Unterschieden soll aber auch das Gemeinsame im Grund- und Leistungsfach betont werden: Die unterrichtliche Gestaltung des Fachs Informatik soll – wie in allen anderen Fächern in der Sekundarstufe II auch – wissenschaftspropädeutisch erfolgen. Damit geht sie über das Anspruchsniveau im Wahlfach / Wahlpflichtfach der Sekundarstufe I hinaus. Die Wissenschaftsorientierung muss aber schulgerecht erfolgen. Ziel ist nicht, im Unterricht ein Abbild der Fachwissenschaft zu entwickeln, sondern wesentliche Probleme, Lösungsansätze, Arbeitsmethoden, Ergebnisse und Theorien des Fachs sowie deren Bedeutung im gesamtgesellschaftlichen Kontext schülerorientiert zu vermitteln.

2 Grundfach

2.1 Information und ihre Darstellung

Computerunterstützte Informationsverarbeitung ist zu einem wesentlichen Bestandteil unserer Gesellschaft geworden. Schülerinnen und Schüler sollen daher lernen, sich in einer Informationsgesellschaft sicher und verständig zu behaupten und deren Weiterentwicklung kritisch mitzugestalten. Dazu sind solide Kenntnisse über die Grundlagen der Informationsverarbeitung erforderlich.

Computerunterstützte Informationsverarbeitung läuft immer in drei Schritten ab: Information muss zunächst geeignet in Form von Daten dargestellt werden; erst diese Daten können maschinell zu neuen Daten verarbeitet werden; durch Interpretation der erzeugten Daten gewinnt man schließlich neue Information. Um diesen Prozess zu verstehen, sollen Schülerinnen und Schüler sich mit verschiedenen Aspekten dieser Schritte beschäftigen. Sie sollen dabei lernen, wie Information einerseits strukturiert, andererseits technisch dargestellt werden kann, wie große Datenbestände in Datenbanken verwaltet werden und welche rechtlichen Fragen hierbei zu beachten sind.

Rechtliche Aspekte sind insofern von besonderer Bedeutung, da viele Jugendliche das Internet als rechtsfreien Raum wahrnehmen. Der Informatikunterricht soll hier entgegenwirken und für einen rechtlich einwandfreien Umgang mit Information sensibilisieren. Aspekte des Urheberrechts sollen ebenso behandelt werden wie Rechte und Pflichten, die sich aus dem „Recht auf informationelle Selbstbestimmung“ ergeben.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Information zur Weiterverarbeitung in Informatiksystemen aufbereiten und sachgerecht Information aus den Verarbeitungsergebnissen gewinnen

Dazu gehört:

- Information adäquat zur Weiterverarbeitung mit dem Computer darstellen
- Binäre Darstellung von Daten erläutern
- Rechtliche Aspekte beim Umgang mit Information beachten
- Datenbanken zur Informationsgewinnung nutzen
- Datenerhebungen unter dem Aspekt Datenschutz bewerten

Information adäquat zur Weiterverarbeitung mit dem Computer darstellen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Unterscheidung zwischen Information und Daten	➢ Anhand von Beispielen herausarbeiten, dass Information immer durch Daten dargestellt werden muss, bevor sie vom Computer verarbeitet oder transportiert werden kann und dass neue Information durch eine Deutung von Daten gewonnen werden kann. ➢ Bewusst machen, dass die Form der Darstellung vom verwendeten Software-Werkzeug abhängt.
Strukturierte Darstellung von Information	➢ Informationsdarstellung im Internet als Fallstudie betrachten. ➢ Die Trennung zwischen Inhalt (Information), Struktur und Formatierung als Grundprinzip herausarbeiten und bei der Darstellung von Information beachten. ➢ Einfache Webseiten mit Hilfe einer Auszeichnungssprache (XHTML) und einer Formatierungssprache (CSS) erstellen. ➢ Wichtige Strukturelemente (wie Titel, Überschrift, Absatz, Verweis, Liste, Bild, Tabelle) zur Darstellung von Information in Hypertexten bewusst machen und zur Strukturierung nutzen. ➢ Die Problematik von unzureichend strukturierten Webseiten bewusst machen (z. B. im Hinblick auf einen barrierefreien Zugang). ➢ Auf entsprechende Strukturierungsmöglichkeiten in anderen Bereichen hinweisen (z. B. SVG als Standard zur Beschreibung skalierbarer Vektorgrafiken).

Darstellung mit formalen Sprachen	➢ Mit Hilfe geeigneter Validierer die syntaktische Korrektheit von Webseiten überprüfen lassen und dabei herausarbeiten, dass die zur Erstellung von Webseiten benutzte Auszeichnungssprache (XHTML) eine formale Sprache ist, bei der strenge Syntaxregeln beachtet werden müssen. ➢ Zur weiteren Vertiefung einen Einblick in die Festlegung von Auszeichnungssprachen mit Hilfe von Dokumententypdefinitionen in XML gewinnen als Möglichkeit zur exakten Definition der Syntax einer Auszeichnungssprache erkunden. ➢ Syntax und Semantik als wesentliche Aspekte einer (formalen) Sprache herausstellen und in verschiedenen Kontexten (wie Informationsdarstellung mit Webseiten; Erstellung von Programmen) thematisieren.

Binäre Darstellung von Daten erläutern	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Bit und Byte	➢ Die Bedeutung binärer Daten für die technische Verarbeitung klären ➢ Einfache technische Realisierungen der Bit-Werte 0 und 1 aufzeigen
Binärdarstellung von Zahlen	➢ Darstellung von Zahlen im Dualsystem besprechen ➢ Dualzahlen hexadezimal darstellen ➢ Einfache Umwandlungsalgorithmen entwickeln
Binärdarstellung von Zeichen	➢ Darstellung mit standardisierten Codes (z.B. ASCII, Unicode, UTF-8) aufzeigen und mit Hilfe eines Hex-Editors verdeutlichen.
Binärdarstellung von Bildern, Tönen, Filmen	➢ Die binäre Darstellbarkeit von Bildern, Tönen und Filmen exemplarisch besprechen (z.B. das Dateiformat PBM als Dateiformat zur Speicherung von Bildern). ➢ Die Abhängigkeit des Speicherbedarfs von der Darstellung klären (z.B. die binäre Version von PBM nutzen). ➢ Den Unterschied zwischen Pixel- und Vektorgrafik klären.

Rechtliche Aspekte beim Umgang mit Information beachten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Persönlichkeitsrechte	➢ Auf die Rechte und Pflichten bei der Verwendung von persönlichen Daten eingehen.
Urheberrecht	➢ Schülerinnen und Schüler für einen rechtlich einwandfreien Umgang mit Information sensibilisieren. ➢ Anhand von Fallstudien Nutzungsrechte bei Software Texten, Bildern, Musik, Filmen klären und bei vergleichbarer Nutzung beachten. ➢ Lizenzierungsmöglichkeiten diskutieren z.B. GPL, Freeware, Shareware, ... ➢ Anhand von Fallstudien auf die von Film- und Musikindustrie eingesetzten Kopierschutzprogramme eingehen.

Datenbanken zur Informationsgewinnung nutzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Bedeutung von Datenbanken	➢ Anhand typischer Beispiele aus der Erfahrungswelt der Schülerinnen und Schüler (z.B. Schulverwaltung, Mobilfunkanbieter, öffentliche Verwaltung, Banken, Einkaufen im Internet, ...) bewusst machen, dass Datenbestände in sehr vielen Bereichen des täglichen Lebens in Datenbanken verwaltet werden. ➢ Den Unterschied in den Verarbeitungsmöglichkeiten zwischen einer nicht-elektronischen (z.B. Telefonbuch, Karteikasten) und einer elektronischen Datensammlung (z.B. Telefon-CD, Datenbank) klären. ➢ Bewusst machen, dass Daten oft auf Dauer gespeichert werden, global verfügbar sind, sehr schnell verarbeitet und auch vernetzt werden können.
Informationsdarstellung mit verknüpften Tabellen	➢ Die Struktur eines Tabellenmodells herausarbeiten: Tabellenschema, Datensätze, atomare Dateneinträge ➢ Redundanz und Inkonsistenz von Daten als Problem erkennen. ➢ Aufteilung der Daten in verknüpfte Tabellen als Lösungsansatz erfahren: Verknüpfung von Tabellen mit Hilfe von Schlüsselattributen

Erstellung von Abfragen mit einer Abfragesprache	➢ Einfache Abfragen direkt mit der QbE-Methode (Query by Example) erstellen. ➢ Grundoperationen zur Auswertung einer Abfrage an ein Tabellenmodell herausarbeiten und als zentrale Elemente der Abfragesprache SQL herausstellen: Auswahl von Datensätzen über eine Selektion, Auswahl von Attributen über eine Projektion, Verknüpfen von Tabellen über eine Produktbildung ➢ SQL nutzen, um einfache Abfragen an Datenbanken zu formulieren. ➢ Vorteile einer Standard-Abfragesprache wie SQL besprechen.

Datenerhebungen unter dem Aspekt Datenschutz bewerten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Sammlung personenbezogener Daten	➢ Klären, wer zu welchem Zweck personenbezogene Daten erheben muss (z. B. zur Erstellung der Handy-Rechnung) bzw. sammeln möchte (z. B. für Werbezwecke). ➢ Anhand von Beispielen besprechen, wie solche Daten erhoben bzw. gesammelt werden (z.B. Anmeldeformulare, Preisausschreiben, Kundenkarte)
Missbrauch personenbezogener Daten	➢ Anhand von Fallstudien aufzeigen, wie mit personenbezogenen Daten Missbrauch betrieben werden kann – insbesondere, wenn Datenbestände aus verschiedenen Quellen zusammengeführt werden. ➢ Die Preisgabe personenbezogener Daten problematisieren (z. B. bei Bestellformularen, Preisausschreiben, Kundenkarten)
Schutz personenbezogener Daten	➢ Das Recht auf informationelle Selbstbestimmung sowie Rechte von Betroffenen besprechen (z.B. Recht auf Auskunft, Einsicht, Berichtigung, Sperrung und Löschung) ➢ Anhand von Beispielen aufzeigen, dass das Recht auf Datenschutz unter bestimmten Umständen eingeschränkt ist.

Additum: Formale Sprachen und Automaten zur Sprachbeschreibung und Spracherkennung nutzen	
Inhalte	**Hinweise für eine mögliche Umsetzung**
Sprachbeschreibung	➢ Gemeinsamkeiten und Unterschiede von natürlichen und formalen Sprachen herausarbeiten. ➢ Häufig benutzte Techniken zur Syntaxfestlegung wie Syntaxdiagramme, reguläre Ausdrücke, Grammatiken erkunden und vergleichen. ➢ Einen Einblick in die XML-Sprachbeschreibung mit Hilfe von Dokumententypdefinitionen gewinnen.
Spracherkennung mit Automaten	➢ Automatenmodelle (endlicher Automat als Akzeptor, Kellerautomat) zur Syntaxanalyse entwickeln ➢ Die Modelle mit geeigneten Simulationsprogrammen (z.B. JFlap) verdeutlichen. ➢ Den praktischen Nutzen abstrakter Modelle bei der Entwicklung von Spracherkennungswerkzeugen wie Scanner und Parser erfahren.
Sprachklassen	➢ Es genügt, einen vertiefenden Einblick in reguläre und kontextfreie Sprachen zu gewinnen und hier Zusammenhänge zu entsprechenden Automatenmodellen zu erkennen. ➢ Exemplarisch die Bedeutung von Theoriebildung in der Informatik diskutieren.

2.2 Aufbau und Funktionsweise eines Rechners

Um die Möglichkeiten und Grenzen des Computers einschätzen zu können, ist auch ein grundlegendes Verständnis der Funktionsweise eines Computers unerlässlich. Auch wenn Rechner technisch ständig weiter entwickelt werden, sind wesentliche Grundprinzipien über viele Rechner-Generationen stabil geblieben. Das Verständnis dieser Grundprinzipien trägt wesentlich zur Entmystifizierung dieser komplexen Maschine bei.

Schülerinnen und Schüler sollen Aufbau und Funktionsweise eines Rechners sowohl von der Softwareseite als auch von der Hardwareseite kennen lernen. Die Wirkungsweise eines Rechners erschließt sich als Zusammenspiel verschiedener Ebenen, die von der Übersetzung von Programmen einer Hochsprache auf Maschinenebene bis zur Ausführung von Maschinenprogrammen im Prozessor reichen. Die eigentliche Hardware nimmt dabei einen verhältnismäßig geringen Teil ein. Das abstrakte Erfassen der Komponenten eines Rechners in ihrer Funktion und ihrem Zusammenspiel - die sogenannte Rechnerarchitektur - ist dabei wichtiger als die konkrete digitaltechnische Realisierung, die dem Wahlbereich vorbehalten ist.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Aufbau und Funktionsweise eines Rechners in ihren Grundlagen erklären

Dazu gehört:

- Sprachebenen und Phasen eines Übersetzungsvorgangs erläutern
- Komponenten eines Rechners in ihrem Zusammenwirken erläutern

Sprachebenen und Phasen eines Übersetzungsvorgangs erläutern	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Assembler- und Maschinensprache	➢ Eine einfache Assemblersprache (z. B. Registermaschinensprache) einführen und exemplarisch zur Steuerung eines Modell-Rechners anwenden. ➢ Den Zusammenhang zwischen Assemblerprogrammen und Maschinenprogrammen klären. ➢ Die Unzulänglichkeit einer Maschinensprache für größere Programmieraufgaben diskutieren.
Automatisierbarkeit von Übersetzungsvorgängen	➢ Übersetzungsvorgänge in einfachen Fällen (z.B. für Kontrollstrukturen) per Hand durchspielen. ➢ Die Automatisierbarkeit eines Übersetzungsvorgangs anhand einfacher Beispiele (z.B. einer Zuweisung) einsichtig machen. ➢ Den gesamten Vorgang in eine Analyse- und Synthesephase zerlegen. Die jeweiligen Aufgaben mit Hilfe von Scanner, Parser und Codegenerator verdeutlichen.
Übersetzung und Interpretation von Hochsprachen	➢ Verschiedene Ansätze zur Ausführung einer Hochsprache besprechen und anhand von Beispielen verdeutlichen ➢ Übersetzung in die Zielsprache, z. B. bei Delphidirekte ➢ Interpretation, z. B. bei Skriptsprachen ➢ Übersetzung in eine Zwischensprache mit anschließender Interpretation, z. B. bei Java. ➢ Vor- und Nachteile der verschiedenen Ansätze diskutieren (z.B. Plattformunabhängigkeit, Geschwindigkeit).

Komponenten eines Rechners in ihrem Zusammenwirken erläutern	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Komponenten eines Rechners	➢ Einen ersten Einblick in den Aufbau von Rechnern durch Öffnen eines Rechners gewinnen. ➢ Komponenten eines von-Neumann-Rechners und ihr Zusammenwirken im Modell (Hardware-Modell und/ oder Software-Simulation) erkunden. ➢ Die Aufgaben der Komponenten differenziert beschreiben: Rechenwerk, Speicher, Befehlsregister, Programmzähler, Bussystem, Steuerwerk.

Befehlszyklus/ Fundamentalzyklus	➢ Die Arbeitsweise eines von-Neumann-Rechners mit Phasen beschreiben. ➢ Programmierbarkeit als zentrale Eigenschaft eines Rechners herausstellen und ihre Realisierung im von-Neumann-Rechner erklären.
Arbeitsgeschwindigkeit	➢ Parameter für die Arbeitsgeschwindigkeit wie Taktfrequenz, Taktanzahl zur Ausführung eines Befehls klären.

Additum: Komponenten eines Rechners mit Digitaltechnik realisieren	
Inhalte	**Hinweise für eine mögliche Umsetzung**
Logische Grundlagen	➢ Logische Grundoperationen (AND, OR, NOT, NAND, NOR) mit ihren Wahrheitstabellen erarbeiten. ➢ Mit realen ICs und/oder mit Simulationsprogrammen (z.B. Hades, LogicSim) experimentieren. ➢ Vor- und Nachteile von digitaler und analoger Darstellung abwägen. ➢ Ausgewählte Gesetze durch Wahrheitstabellen und/oder Schaltungen nachweisen.
Schaltnetze	➢ Schaltnetze mit konjunktiver/disjunktiver Normalform systematisch erstellen, logische Terme durch Umformungen exemplarisch vereinfachen. ➢ Und-Gatter als Datentor einsetzen ➢ Halbaddierer, Volladdierer, n-Bit Volladdierer entwickeln und testen.
Schaltwerke	➢ RS-Flipflop als Grund-Flipflop herleiten ➢ Mit taktzustandgesteuertem Flipflop einen 1-Bit-Speicher aufbauen. ➢ Mit flankengesteuerten Master-Slave-Flipflops Vorwärts- und Rückwärtszähler aufbauen.
Buskonzept	➢ Bus als gemeinsames Kommunikationsmedium erfahren ➢ Datentransfer an einem Bus mit Tristate-Gatter exemplarisch aufzeigen.

2.3 Kommunikation in Rechnernetzen

Rechnernetze und das Internet gehören mittlerweile zum Alltag der meisten Schülerinnen und Schüler. Im Informatikunterricht soll ein Verständnis der Kommunikation in Netzen - insbesondere der im Internet - erreicht werden.

Die Strukturen von Rechnernetzen können anhand historischer Kommunikationssysteme und alltäglicher Kommunikationssituationen analysiert und beschrieben werden. Im Unterricht werden dabei Lösungsansätze für die notwendigen Dienste und Protokolle herausgearbeitet.

Des Weiteren spielen Fragen zur Datensicherheit bei der Übertragung von Daten eine wesentliche Rolle. Im Unterricht werden die dazugehörigen Sicherheitsprobleme und kryptologische Verfahren als ihre Lösung herausgearbeitet. Hierdurch werden die Schülerinnen und Schüler für Gefahren innerhalb ihrer alltäglichen elektronischen Kommunikation sensibilisiert

Aspekte kryptologischer Verfahren sollen durch geeignete Werkzeuge verdeutlicht werden. Darüber hinaus sollen die Schülerinnen und Schüler mit der Anwendung kryptologischer Verfahren für die alltägliche Kommunikation vertraut gemacht werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Kommunikation in Rechnernetzen in ihren Grundlagen erklären

Dazu gehört:

- Grundstrukturen von Kommunikationssystemen analysieren und beschreiben
- Kommunikation in Rechnernetzen erläutern und am Beispiel des Internet verdeutlichen
- Datensicherheit unter Berücksichtigung kryptologischer Verfahren erklären und beachten

Grundstrukturen von Kommunikationssystemen analysieren und beschreiben	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Elemente von Kommunikationssystemen	– Einfache Kommunikationsvorgänge experimentell durchführen (z.B. Morsen, Lichtsignale, Handzeichen) und dabei die Konzepte „Sender“, „Empfänger“, „Nachricht“ und „Protokoll“ erarbeiten. – Protokolle als Summe aller Vereinbarungen zwischen Sender und Empfänger zur Abwicklung von Kommunikationsvorgängen beschreiben.
Eigenschaften von Kommunikationssystemen	– Anhand historischer Verfahren (z.B. optischer Telegrafie) und Kommunikationsvorgängen aus dem Alltag (z.B. Telefonieren, Briefversand) Eigenschaften (z.B. Schnelligkeit von Kommunikationssystemen herausarbeiten. – Den Unterschied zwischen verbindungsloser und verbindungsorientierter bzw. bestätigter und unbestätigter Kommunikation verdeutlichen.

Kommunikation in Rechnernetzen erläutern und am Beispiel des Internet verdeutlichen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Grundprobleme und Lösungsansätze	– Grundprobleme unterschiedlicher Systeme erkennen und exemplarisch einen Lösungsansatz erarbeiten: – Bitübertragung zwischen zwei Rechnern (einfache serielle Übertragung von Datenpaketen) – Erkennung von Bitübertragungsfehlern (Prüfsumme, zyklische Redundanzüberprüfung CRC) – Reaktion auf Übertragungsfehler (automatische Wiederholungsanfrage, Quittungsbetrieb) – Datenkollision bei Systemen mit mehreren Sendern und Empfängern (weitere Informationen unter den Stichworten ALOHA, CSMA/CD) – Wegewahl in einem Netz von Netzen (Routingtabelle, Routing Information Protocol)

Schichtenarchitektur	➢ Anhand eines einfachen Schichtenmodells die Vorteile einer geschichteten System-Architektur besprechen ➢ Den Unterschied zwischen Diensten und Protokollen einer Schicht erläutern.
Dienste und Protokolle des Internet	➢ Kommunikationsvorgänge mit der Client-Server-Struktur beschreiben. ➢ Einen Überblick über aktuelle Dienste und Protokolle des Internet verschaffen. ➢ Exemplarisch ein Protokoll (z.B. HTTP, POP, SMTP, FTP) mit Hilfe eines Terminalprogramms und mit Hilfe von RFC-Auszügen untersuchen. ➢ Optional: Mit Hilfe von vorhandenen Klassen, die eine TCP/IP-Verbindung herstellen (Sockets), eine Anwendung programmieren.

Datensicherheit unter Berücksichtigung kryptologischer Verfahren erklären und beachten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Sicherheitsziele	➢ Sicherheitsprobleme bei Kommunikationsvorgängen im Alltag aufzeigen. ➢ Die Brisanz von Sicherheitsproblemen bei elektronischer Kommunikation (z.B. Chat, E-Mail, Online-Banking, E-Vote) herausstellen und diskutieren. ➢ Vertraulichkeit, Authentizität, Integrität, Verbindlichkeit als Sicherheitsziele herausarbeiten.
Moderne Verfahren zur Verschlüsselung und Signierung	➢ Historische Verfahren nur als Anknüpfungspunkte an das Thema „Verschlüsselung“ nutzen. ➢ Das Grundprinzip asymmetrischer Verfahren erarbeiten. Dabei das Prinzip der Einwegfunktion besprechen und an Beispielen verdeutlichen – die komplexen mathematischen Hintergründe der benutzten Verfahren allenfalls kurz thematisieren. ➢ Mit aktuellen Werkzeugen ver-/ entschlüsseln und signieren (z.B. GnuPG).
Sicherheitsinfrastruktur	➢ Probleme zur Sicherheitsinfrastruktur besprechen: Schlüsselvergabe, Zertifizierung von Schlüsseln. ➢ Die Grundlagen einer Sicherheitsinfrastruktur durch Austausch und gegenseitige Signierung von Schlüsseln innerhalb des Kurses aufzeigen.

Additum: Verfahren der Kommunikation in Rechnernetzen realisieren	
Inhalte	**Hinweise für eine mögliche Umsetzung**
Kommunikation in Rechnernetzen	➢ Einfache Bit-Übertragung programmtechnisch realisieren - Bitübertragung zwischen zwei Rechnern (einfache serielle Übertragung von Datenpaketen) - Erkennung von Bitübertragungsfehlern (Prüfsumme, CRC) - Reaktion auf Übertragungsfehler (automatische Wiederholungsanfrage, Quittungsbetrieb) - Datenkollision bei Systemen mit mehreren Sendern und Empfängern (ALOHA, CSMA/CD) - Wegewahl in einem Netz von Netzen (Routingtabelle, z.B. Routing Information Protocol)
Dienste und Protokolle des Internet	➢ Mit Hilfe von vorhandenen Klassen, die eine TCP/IP-Verbindung herstellen (Sockets), eine Anwendung programmieren.
Moderne Verfahren zur Verschlüsselung und Signierung	➢ Das Grundprinzip des RSA-Verfahrens erarbeiten. ➢ Berechnungen von öffentlichen und geheimen Schlüsseln für einfache Zahlenbeispiele durchführen und Kodierungen bzw. Dekodierungen nachvollziehen.

2.4 Algorithmisches Problemlösen

Algorithmen werden in der Informatik in vielen Bereichen zur Lösung von Problemstellungen benötigt - letztlich basiert jede Computeranwendung auf Algorithmen.

Um selbst Probleme algorithmisch lösen zu lernen, sollen Schülerinnen und Schüler sich mit Grundstrukturen von Algorithmen, Strategien zur Entwicklung von Algorithmen sowie Eigenschaften und Grenzen von Algorithmen auseinandersetzen.

Die Entwicklung von Algorithmen sollte stets von praktischen Problemstellungen ausgehen und in der Regel auch eine Implementierung in einer Programmiersprache einschließen. Eine eigenständige Behandlung von Grundstrukturen ist denkbar, der gesamte Themenbereich soll aber nicht als geschlossene Unterrichtsreihe behandelt, sondern mit anderen Bereichen vernetzt werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Algorithmische Problemlösungen entwickeln und die Grenzen dieser Methode einschätzen

Dazu gehört:

- Die Bedeutung der algorithmischen Datenverarbeitung einschätzen
- Algorithmische Grundstrukturen beherrschen
- Algorithmen entwickeln und implementieren
- Grenzen der algorithmischen Datenverarbeitung einschätzen

Die Bedeutung der algorithmischen Datenverarbeitung einschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Algorithmusbegriff	➢ Bewusst machen, dass jede automatisierte Verarbeitung von Daten mithilfe des Computers auf der Grundlage präziser Verarbeitungsvorschriften erfolgt ➢ Den Algorithmusbegriff klären; Anforderungen an Algorithmen formulieren
Bedeutung von Algorithmen früher und heute	➢ Anhand von Beispielen aufzeigen, dass es Algorithmen schon gab, bevor es den Computer gegeben hat (z.B. Algorithmen in der Mathematik) ➢ Die besondere Bedeutung, die Algorithmen durch die Entwicklung des Computers gewonnen haben, bewusst machen (Möglichkeit der schnellen automatisierten Ausführung)

Algorithmische Grundstrukturen beherrschen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Variablenkonzept, Datentypkonzept	➢ Grundvorstellungen zur Datenspeicherung mit Variablen entwickeln. ➢ Zuweisung als Grundoperation zur Veränderung von Variablenwerten verwenden. ➢ Grundlegende Datentypen thematisieren (eine ausführliche und vertiefende Behandlung spezieller Datentypen ist nicht intendiert) ➢ Die Grundidee einer Typisierung herausarbeiten.
Kontrollstrukturen	➢ Die Rolle der Kontrollstrukturen Sequenz, Fallunterscheidung, Wiederholung bei der Ablaufmodellierung bewusst machen. ➢ Kontrollstrukturen vielfältig zur Ablaufmodellierung verwenden. ➢ Wahrheitswerte und logische Verknüpfungen als Grundlage von Bedingungen herausstellen.
Datenstrukturen	➢ Reihung als einfache Datenstruktur einführen und vielfältig verwenden. ➢ Weitere Datenstrukturen (wie Liste, Stapel) bei Bedarf problemorientiert thematisieren.

Prozedurkonzept, Parameterkonzept	➢ Prozeduren und Funktionen als Mittel zur modularen Darstellung von Algorithmen einführen und vielfältig nutzen. ➢ Den Unterschied zwischen Prozeduren und Funktionen klären. ➢ Die Bedeutung von Parametern (formale, aktuelle) klären. ➢ Die Signatur einer Prozedur jeweils genau spezifizieren (Anzahl der Parameter, Typ der Parameter, ...)
Darstellungsformen	➢ Umgangssprache zur informellen Beschreibung von Abläufen nutzen. ➢ Flussdiagramme zur Veranschaulichung von Verzweigungen und Wiederholungen einsetzen. ➢ Struktogramme zur strukturbetonten Darstellung von Algorithmen benutzen. ➢ Mit geeigneten Editoren die Darstellung von Algorithmen unterstützen.

Algorithmen entwickeln und implementieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Problemanalyse	➢ Problemsituationen analysieren (Anfangszustand, gewünschter Endzustand) und anhand konkreter Beispiele verdeutlichen. ➢ Den Zusammenhang zwischen Anfangszustand und Endzustand möglichst genau beschreiben.
Strategien zur Entwicklung algorithmischer Problemlösungen	➢ Das Schema „Eingabe - Verarbeitung - Ausgabe“ einsichtig machen und zur Strukturierung von Systemen nutzen ➢ Das Top-Down-Verfahren und das Bottom-Up-Verfahren in geeigneten Fällen anwenden und die zu Grunde liegenden Strategien herausstellen. ➢ Typische algorithmische Lösungsmuster besprechen (z. B. Vertauschen von zwei Werten) ➢ Abläufe anhand konkreter Problemfälle durchspielen und die Ablaufmuster schrittweise mit Hilfe algorithmischer Grundkonzepte modellieren. ➢ Raum für kreative Lösungen gewähren.

Problemlösen mit Standardalgorithmen	➢ Standardalgorithmen zur Lösung von Standardproblemen (wie z. B. schnelles Sortieren) nutzen. ➢ Standardalgorithmen der Fachliteratur entnehmen, die Grundideen herausstellen und informell darstellen. ➢ Standardalgorithmen an spezielle Problemsituationen anpassen.
Korrektheit	➢ Korrektheit eines Algorithmus als Problem thematisieren ➢ Mithilfe gezielter Tests das Korrektheitsverhalten eines Algorithmus untersuchen. ➢ Das Verhalten eines Algorithmus in Einzelfällen durch manuell angefertigte Ablaufprotokolle untersuchen.
Effizienz	➢ Die Bedeutung von Zeit- und Speicheraufwand bei der Entwicklung von Algorithmen klären. ➢ Exemplarisch Algorithmen hinsichtlich ihres Laufzeitverhaltens vergleichen. ➢ Den Zeitaufwand dabei experimentell oder durch Abschätzungen ermitteln.
Implementierung von Algorithmen	➢ Die Darstellung algorithmischer Konzepte in der gewählten Programmiersprache thematisieren, soweit sie für die Umsetzung von Algorithmen erforderlich ist. ➢ Typische Fehlermuster besprechen (z. B. Unterschied zwischen Vergleich und Wertzuweisung).

Grenzen der algorithmischen Datenverarbeitung einschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Praktische Grenzen	➢ Anhand eines geeigneten Problems (z. B. Primfaktorzerlegung) experimentell erfahren oder durch Überlegungen herausfinden, dass es Probleme gibt, die zwar algorithmisch lösbar sind, deren Lösungen aber praktisch nicht genutzt werden können. ➢ Exponentiellen Aufwand als praktische Grenze erfahren.
Prinzipielle Grenzen	➢ Termination als Problem anhand geeigneter Algorithmen (z.B. Goldbach-Eigenschaft, Collatz-Folge, Primzahl-Zwillinge) verdeutlichen. ➢ Die Grenze der algorithmischen Problemlösemethode anhand des Halteproblems verdeutlichen (es reicht, die wesentlichen Ergebnisse und ihre Konsequenzen ohne mathematische Fundierung zu thematisieren).

Additum: Grenzen algorithmisch arbeitender Systeme theoretisch aufzeigen	
Inhalte	**Hinweise für eine mögliche Umsetzung**
Berechnungsmodell	➢ Die Turingmaschine als Berechnungsmodell einführen. ➢ Einfache Berechnungsprobleme mit Turingmaschinen lösen. ➢ Die Menge der Turing-berechenbaren Funktionen als abzählbar einsehen und die Existenz einer nicht-Turing-berechenbaren Funktion belegen. ➢ Fleißige-Biber-Turingmaschinen kennen lernen und die Rado-Funktion als Beispiel für eine nicht-Turing-berechenbare Funktion erleben.
Church-Turing-These	➢ Alternative Berechnungsmodelle aufzeigen. ➢ Die Äquivalenz der verschiedenen Modelle plausibel machen bzw. mitteilen.

2.5 Informatische Modellierung

Modellierung spielt innerhalb der Informatik eine zentrale Rolle. Informatiksysteme können aufgrund der zu bewältigenden Komplexität nicht ad hoc entwickelt werden. Man benötigt Baupläne, die den betreffenden Gegenstandsbereich sowie das gewünschte Systemverhalten vereinfachend und strukturgetreu beschreiben. Modellierung soll daher auch die Arbeitsweise im Unterricht prägen. Mit Hilfe von Modellen soll einerseits die Entwicklung von Problemlösungen systematisiert, andererseits das Verständnis von Informatiksystemen durch eine vereinfachende Darstellung unterstützt werden. Der Lehrplan sieht zwei Modellierungsansätze zur intensiveren Bearbeitung im Grundfach vor – zustandsbasierte und objektorientierte Modellierung.

Zustandsbasierte Modellierung erweist sich als einfacher und in vielen Bereichen der Informatik anwendbarer Modellierungsansatz, der im gesamten Informatikunterricht immer dann zur Analyse und Entwicklung von Systemen genutzt werden sollte, wenn das Verhalten eines Systems nicht nur von äußeren Eingaben oder Ereignissen abhängt, sondern auch von inneren Zuständen, die das System in Abhängigkeit von den Eingaben bzw. Ereignissen durchläuft.

Objektorientierte Konzepte nehmen in der Software-Entwicklung derzeit eine herausragende Stellung ein. Gängige Programmiersysteme bieten in der Regel umfangreiche Klassenbibliotheken an, mit deren Hilfe Objekte erzeugt werden können, die eine Vielzahl von Funktionalitäten bereit stellen. Im Informatikunterricht sollen die Grundideen und Grundlagen der Objektorientierung in didaktisch reduzierter Form behandelt werden, um vorgegebene Klassen nutzen zu können und die hiermit verbundenen Modularisierungsprinzipien kennen zu lernen.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Informatische Modelle entwickeln und implementieren

Dazu gehört:

- Zustandsbasierte Modelle zu einfachen Problembereichen entwickeln
- Grundideen und Grundkonzepte der objektorientierten Modellierung erklären
- Objektorientierte Modelle zu einfachen Problembereichen entwickeln und implementieren

Zustandsbasierte Modelle zu einfachen Problembereichen entwickeln	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Basiskonzepte zustandsbasierter Modelle	➢ Anhand einfacher und aus dem Alltag bekannter Systeme die Basiskonzepte der zustandsbasierten Modellierung einführen: Zustand, Anfangszustand, Endzustände, Zustandsübergang.
Darstellung zustandsbasierter Modelle	➢ Zustandsgraph und Zustandstabelle zur Beschreibung zustandsbasierter Modelle einführen und nutzen.
Systembeschreibung mit zustandsbasierten Modelle	➢ Zustandsbasierte Modellierung vielfältig zur Beschreibung von Informatiksystemen nutzen. ➢ Ereignisgesteuerte Systeme als zustandsbasierte Systeme interpretieren (Ereignis als auslösende Aktion eines Zustandsübergangs; Ereignisbehandlung als hiermit verbundene ausgelöste Aktion)
Simulation zustandsbasierter Modelle	➢ Zustandsbasierte Modelle mit Hilfe von Werkzeugen simulieren. ➢ Programme zur Simulation zustandsbasierter Modelle erstellen und nutzen.

Grundideen und Grundkonzepte der objektorientierten Modellierung erklären	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Basiskonzepte der Objektorientierung: Klasse, Objekt, Nachricht	➢ Vorgegebene Klassenbibliotheken bei einer Problembearbeitung nutzen (z. B. zur Gestaltung einer Benutzungsoberfläche) und analysierend dabei erste Vorstellungen zum Objekt- und Klassenkonzept entwickeln: – Objekt als Einheit aus Attributen und Methoden, das für die Erledigung bestimmter Aufgaben zuständig ist – Objekt als aktive und autonome Programmeinheit mit eigener Identität, das durch eine Nachricht veranlasst werden kann, bestimmte Aufgaben zu erledigen. – Klasse als Bauplan für Objekte, der für die Erzeugung von Objekten benötigt wird Schnittstelle der Klasse als Summe aller Informationen, die zur Benutzung der Klassen benötigt werden.

	➢ Eigene Klassen zur Problembearbeitung entwickeln und dabei das Verständnis des Objekt- und Klassenkonzepts vertiefen: – Objekt als abstrahierende Entsprechung eines Gegenstands aus dem Problembereich ➢ Klasse als Objekttyp, die Struktur und Verhalten der zu erzeugenden Objekte genau festlegt
Basiskonzepte der Objektorientierung: Beziehung	➢ Grundvorstellungen zum Beziehungskonzept entwickeln: – Beziehung zwischen Objekten als Grundlage dafür, dass mehrere Objekte gemeinsam eine komplexere Aufgabe erledigen – Herstellen von Beziehungen durch Referenzen.
Prinzipien der Objektorientierung	➢ Die Relevanz objektorientierter Konzepte mit Prinzipien der Informatik verdeutlichen, dabei insbesondere folgende Aspekte diskutieren: – Klasse als abgeschlossene Einheit (Modularisierung) mit einer eindeutigen Schnittstelle, die verwendet werden kann, ohne die Details ihrer Realisierung zu kennen (Geheimnisprinzip) – Klasse als Programmierbaustein, der vielfältig verwendet werden und daher abstrahierend entworfen werden sollte (Abstraktion, Wiederverwendbarkeit)

Objektorientierte Modelle zu einfachen Problembereichen entwickeln und implementieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Entwicklung objektorientierter Modelle	➢ Die Problembereiche zunächst sehr einfach und anschaulich wählen, erst nach und nach den Komplexitätsgrad etwas steigern. ➢ Funktionen eines Modells herausstellen und bei der Modellbildung beachten: – Modell als Abbild der Miniwelt – Modell als Vorlage für ein Informatiksystem. ➢ Modelle entsprechend schrittweise ausbauen: – Die Miniwelt analysieren und geeignete Objekte nach Zuständigkeiten konzipieren – Die Aufgaben des geplanten Systems klären und bei der Weiterentwicklung von Modellen beachten. ➢ Elementare Entwurfsmuster herausstellen und bei der Modellierung nutzen, z.B. Trennung von Benutzungsoberfläche und dem Datenmodell (MVC - Konzept)

Darstellung objektorientierter Modelle	➢ Standardisierte Darstellungsformen (UML - Diagramme) einführen und zur Beschreibung von objektorientierten Modellen verwenden. ➢ Modelle mit Hilfe von Entwicklungswerkzeugen (z.B. speziellen Editoren) übersichtlich darstellen und dokumentieren.
Implementierung objektorientierter Modelle	➢ Implementierungsmuster für typische und häufig vorkommende Modellstrukturen erarbeiten und vielfältig nutzen, u. a. – Implementierung von Klassen – Erzeugung und Vernichtung von Objekten. ➢ Typische Fehler besprechen (z.B. Objekt wird benutzt, ist aber nicht erzeugt). ➢ Syntaxregeln der gewählten Programmiersprache thematisieren; Implementierungsdetails mit einem Hilfesystem nachschlagen ➢ Verfahren zur systematischen Fehlersuche erarbeiten (Fehler mit einem Debugger oder durch gezielt eingesetzte Ausgabeanweisungen lokalisieren).

Additum: Komplexere objektorientierte Modelle entwickeln und implementieren	
Inhalte	**Hinweise für eine mögliche Umsetzung**
Beziehung	➢ Verschiedene Beziehungsarten unterscheiden: Hat-Beziehung, Kennt - Beziehung
Vererbung	➢ Vererbung zur Generalisierung oder Spezialisierung von Klassen nutzen ➢ Vererbung bei geeigneten Anwendungsbeispielen zur Verbesserung der Modellstruktur (z. B. Vermeidung von Code-Duplizierung) einsetzen. ➢ Vor- und Nachteile von Klassenhierarchien mit Vererbung diskutieren.

2.6 Software-Entwicklung

Software-Entwicklung ist eine komplexe Aufgabe, an der üblicherweise mehrere Personen beteiligt sind und die zur Bewältigung gut geplantes Vorgehen erfordert. Im Rahmen von schulgerechten Software-Entwicklungsprojekten lassen sich erste Einblicke in Arbeitsprozesse gewinnen, die typisch für viele Bereiche in Wirtschaft und Technik sind. Schülerinnen und Schüler sollen hierbei u. a. erfahren, wie man komplexere Prozesse strukturiert und die Vorgehensweise organisiert, wie man im Team gemeinsam Teilaufgaben bearbeitet und dass man Verantwortung für das Gelingen des Projektes und für das entwickelte Produkt übernehmen muss. Bereits kleinere Entwicklungsprojekte zeigen zudem sehr schnell, wie schwierig es ist, zuverlässig funktionierende und benutzergerechte Software zu erstellen und helfen so, Anforderungen an Software-Qualität realistisch einzuschätzen.

Die Entwicklung kleinerer Software-Produkte kann zudem zum Verständnis vieler informatischer Konzepte beitragen. Indem man die Behandlung fundamentaler Konzepte mit passenden Anwendungsentwicklungen vernetzt, wird dieser Gegenstandsbereich durch die intensivere Beschäftigung vertieft.

Um Software-Entwicklungsprojekte erfolgreich zu gestalten, sollten die zu bewältigenden Aufgaben eher klein und überschaubar gehalten werden. Mehrere kleinere Projekte sind daher in der Regel einem einzigen größeren Projekt vorzuziehen. Hierdurch können auch die zur erfolgreichen Durchführung erforderlichen Kompetenzen gezielter weiterentwickelt werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Software verantwortungsbewusst, systematisch und kooperativ entwickeln.

Dazu gehört:

- Gütekriterien bei der Entwicklung von Software kennen und beachten
- Software-Entwicklungsprozesse systematisch durchführen
- Ein Software-Entwicklungs-Projekt organisieren

Qualitätsmerkmale für Software kennen und beachten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Qualität von Software	➢ Anhand „berühmter Bugs" verdeutlichen, dass auch kommerzielle Software in der Regel Fehler enthält und sich gelegentlich unsicher verhält. ➢ Bei der Entwicklung eigener Softwareprodukte exemplarisch Qualitätsmerkmale beachten (z.B. Benutzerfreundlichkeit, Gestaltung des Quelltextes).
Verantwortlichkeit für Software	➢ Anhand von Fallbeispielen (z.B. Versagen medizinischer Software) die Verantwortlichkeit für Folgen von Software problematisieren. ➢ Ethik-Richtlinien diskutieren.

Software-Entwicklungsprozesse systematisch durchführen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Entwicklungsschritte: Anforderungsanalyse, Modellierung, Implementierung, Testen	➢ Die Vorgehensweise bei der Entwicklung von Software reflektieren und planen. ➢ Bei Entwicklungsprozessen exemplarisch einen der folgenden Schritte vertiefen: – Die Anforderungen ermitteln (z. B. über Fallstudien) und mit einem Pflichtenheft dokumentieren – Modellierungsansätze passend auswählen (zustandsbasiert, objektorientiert, algorithmisch) und das zu entwickelnde System mit Hilfe von Modellen systematisch konzipieren – Die entwickelten Modelle mit der gewählten Programmiersprache implementieren – Alle Funktionalitäten des entwickelten Systems gezielt und isoliert testen
Dokumentation	➢ Schritte exemplarisch dokumentieren (z.B. Pflichtenheft, Schnittstelle einer Klasse, kommentiertes Programm, Testprotokoll). ➢ Werkzeuge (z.B. HTML-Editor, CASE-Tool) bei der Dokumentation einsetzen.

Additum: Ein Software-Entwicklungs-Projekt organisieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Organisationsbereiche: Projektauftrag, Aufgabenverteilung, Arbeitsplan, Absprachen	➢ Ein Projekt durchführen, bei dem eine gewisse Selbstständigkeit in Organisation und Durchführung erforderlich ist. ➢ Die Organisation des Software-Projektes gemeinsam besprechen und in Teilen von Schülerinnen und Schülern übernehmen lassen. ➢ Aufgabenverteilung und Rollenzuweisung klar vereinbaren. ➢ Rahmenbedingungen vorab klären (insbesondere Zeitvorgaben). ➢ Kommunikations- und Kooperationssysteme zum Austausch von Informationen einsetzen. ➢ Sowohl Zwischen- als auch Endergebnisse präsentieren und diskutieren.

3 Leistungsfach

3.1 Sprachen und Automaten

Sprache wird seit jeher zu Kommunikationszwecken benutzt. Mit der Entwicklung von Informatiksystemen gewinnt die Entwicklung von Sprachen, die die Kommunikation zwischen Mensch und Maschine sowie zwischen Maschine und Maschine ermöglicht, eine immer größere Bedeutung. Solche Sprachen sollen im Informatikunterricht nicht nur genutzt, ihre Konstruktion und Verarbeitung soll auch reflektiert und theoretisch fundiert werden. Im Unterricht ist dabei kein „Theoriekurs“ anzustreben. Vielmehr soll durch eine Verzahnung mit praktischen Problemstellungen herausgearbeitet werden, dass eine systematische Beschäftigung mit Möglichkeiten der Sprachbeschreibung und Sprachverarbeitung wesentlich zur Lösung dieser Probleme beiträgt.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Mit künstlichen Sprachen theoretisch fundiert umgehen

Dazu gehört:

- Syntax und Semantik künstlicher Sprachen präzisieren
- Automaten zur Spracherkennung nutzen

Syntax und Semantik künstlicher Sprachen präzisieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Künstliche Sprachen	➢ Die Rolle künstlicher Sprachen in der Informatik besprechen. ➢ Unterschiede zwischen künstlichen und natürlichen Sprachen herausarbeiten. ➢ Anhand konkreter – natürlicher wie künstlicher – Sprachen den Unterschied zwischen Syntax und Semantik klären.
Präzisierung von Syntax	➢ Den Begriff der formalen Sprache präzisieren. ➢ Gängige Techniken zur Präzisierung von Syntax (wie Syntaxdiagramme, Grammatiken, reguläre Ausdrücke) erarbeiten und nutzen. ➢ Die Wahl der im Unterricht behandelten Techniken nach den Intentionen, den bearbeiteten Problemkontexten, den benutzten Werkzeugen etc. treffen. ➢ Abstrakte Konzepte mit geeigneten Simulationsprogrammen (z. B. JFLAP) verdeutlichen. ➢ Syntaxfestlegung in der Praxis erkunden (z. B. Festlegung des Formats von Email-Adressen in der RFC 822 oder Festlegung eines XML-Vokabulars mit einer Dokumententypdefinition).
Präzisierung von Semantik	➢ An einfachen Beispielen klar machen, warum auch die Bedeutung syntaktischer Konstrukte präzise festgelegt werden sollte. ➢ Exemplarisch aufzeigen, wie dies erfolgen kann. Als Lösungsansätze eignen sich der Interpreteransatz, bei dem die Bedeutung der Konstrukte einer Sprache durch das Verhalten eines Interpreters festgelegt wird sowie der Übersetzeransatz, bei dem die Bedeutung von Konstrukten einer neuen Sprache auf die Bedeutung der Konstrukte einer bereits bekannten Sprache zurückgeführt wird.

Automaten zur Spracherkennung nutzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Spracherkennung mit Automaten	➢ Automatenmodelle (endlicher Automat, Kellerautomat, Turingmaschine) zur Spracherkennung entwickeln. ➢ Die Modelle mit geeigneten Simulationsprogrammen (z. B. JFLAP) verdeutlichen. ➢ Den praktischen Nutzen abstrakter Modelle bei der Entwicklung von Spracherkennungswerkzeugen wie Scanner und Parser erfahren.
Sprachklassen und entsprechende Automatenmodelle	➢ Die in der Praxis wichtigsten Sprachklassen (reguläre Sprachen, kontextfreie Sprachen) vertieft behandeln. ➢ Auf weitere Sprachklassen (kontextsensitive Sprachen, allgemeine Sprache) einen Ausblick geben – Chomsky-Hierarchie. ➢ Zusammenhänge zwischen Sprachklassen und entsprechenden Automatenmodelle erarbeiten. ➢ Äquivalenz- und Inklusionseigenschaften anhand von Beispielen verdeutlichen und exemplarisch auch allgemein aufzeigen (z. B. anhand von Klammersprachen). ➢ Die Bedeutung von Theoriebildung in der Informatik diskutieren.

3.2 Aufbau und Funktionsweise eines Rechners

Um die Möglichkeiten und Grenzen des Computers einschätzen zu können, ist auch ein grundlegendes Verständnis der Funktionsweise eines Computers unerlässlich. Auch wenn Rechner technisch ständig weiter entwickelt werden, sind wesentliche Grundprinzipien über viele Rechner-Generationen stabil geblieben. Das Verständnis dieser Grundprinzipien trägt wesentlich zur Entmystifizierung dieser komplexen Maschine bei.

Schülerinnen und Schüler sollen Aufbau und Funktionsweise eines Rechners sowohl von der Softwareseite als auch von der Hardwareseite kennen lernen. Die Wirkungsweise eines Rechners erschließt sich als Zusammenspiel verschiedener Ebenen, die von der Übersetzung von Programmen einer Hochsprache auf Maschinenebene bis zur Ausführung von Maschinenprogrammen im Prozessor reichen. Die eigentliche Hardware nimmt dabei einen verhältnismäßig geringen Teil ein.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Aufbau und Funktionsweise eines Rechners in ihren Grundlagen erklären und bewerten

Dazu gehört:

- Sprachebenen und Phasen eines Übersetzungsvorgangs erläutern
- Komponenten eines Rechners in ihrem Zusammenwirken erläutern und bewerten
- Grundlegende Funktionen eines Betriebssystems angeben

Sprachebenen und Phasen eines Übersetzungsvorgangs erläutern	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Assembler- und Maschinensprache	➢ Eine einfache Assemblersprache (z. B. Registermaschinensprache) einführen und exemplarisch zur Steuerung eines Modell-Rechners anwenden. ➢ Den Zusammenhang zwischen Assemblerprogrammen und Maschinenprogrammen klären. ➢ Die Unzulänglichkeit einer Maschinensprache für größere Programmieraufgaben diskutieren.
Automatisierbarkeit von Übersetzungsvorgängen	➢ Übersetzungsvorgänge in einfachen Fällen (z.B. für Kontrollstrukturen) per Hand durchspielen. ➢ Die Automatisierbarkeit eines Übersetzungsvorgangs anhand von einigen Beispielen einsichtig machen. ➢ Den gesamten Vorgang in eine Analyse- und Synthesephase zerlegen. Die grundlegenden Aufgaben eines Compilers mit Hilfe von Scanner, Parser und Codegenerator verdeutlichen.
Übersetzung und Interpretation von Hochsprachen	➢ Verschiedene Ansätze zur Ausführung einer Hochsprache besprechen und anhand von Beispielen verdeutlichen – Übersetzung in die Zielsprache, z. B. bei Delphi – direkte Interpretation, z. B. bei Skriptsprachen – Übersetzung in eine Zwischensprache mit anschließender Interpretation, z. B. bei Java. ➢ Vor- und Nachteile der verschiedenen Ansätze diskutieren (z.B. Plattformunabhängigkeit, Geschwindigkeit).

Komponenten eines Rechners in ihrem Zusammenwirken erläutern und bewerten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Komponenten eines Rechners	➢ Komponenten eines von-Neumann-Rechners und ihr Zusammenwirken im Modell (Hardware-Modell und/ oder Software-Simulation) erkunden. ➢ Die Aufgaben der Komponenten differenziert beschreiben: Rechenwerk, Speicher, Befehlsregister, Programmzähler, Bussystem.
Befehlszyklus / Fundamentalzyklus	➢ Die Arbeitsweise eines von-Neumann-Rechners mit Phasen beschreiben. ➢ Takt als Mittel zur Sequenzierung der Schritte erkennen ➢ Programmierbarkeit als zentrale Eigenschaft eines Rechners herausstellen und ihre Realisierung im von-Neumann-Rechner erklären.
Funktionsweise eines Steuerwerks	➢ Mikroprogrammierung der Assembler-Befehle erläutern und exemplarisch Befehlsfolgen für eigene Befehle programmieren.
Bewertung des Rechnerkonzepts	➢ Bus als „Flaschenhals“ beim von-Neumann-Rechner erkennen ➢ Den Einfluss von RISC- und CISC-Architekturen diskutieren ➢ Möglichkeiten und Schwierigkeiten der Parallelisierung aufzeigen

Grundlegende Funktionen eines Betriebssystems angeben	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
grundlegende Betriebssystemfunktionen	➢ Grundlegende Betriebssystemfunktionen (z.B. Überwachen der Tastatureingaben, Ausgabe von Daten) in einfachen Modellen (z.B. Tastatur mit nur einer Taste, Ausgabe mit nur einer Leuchtdiode) analysieren. ➢ Eine erste Vorstellung der weiteren Aufgaben eines Betriebssystems (z.B. Laden und Ausführen von Programmen, Verwalten von Prozessen) entwickeln.

3.3 Kommunikation in Rechnernetzen

Rechnernetze und das Internet gehören mittlerweile zum Alltag der meisten Schülerinnen und Schüler. Im Informatikunterricht soll ein Verständnis der Kommunikation in Netzen - insbesondere der im Internet - erreicht werden.

Die Strukturen von Rechnernetzen können anhand historischer Kommunikationssysteme und alltäglicher Kommunikationssituationen analysiert und beschrieben werden. Im Unterricht werden dabei Lösungsansätze für die notwendigen Dienste und Protokolle herausgearbeitet.

Des Weiteren spielen Fragen zur Datensicherheit bei der Übertragung von Daten eine wesentliche Rolle. Im Unterricht werden die dazugehörigen Sicherheitsprobleme und kryptologische Verfahren als ihre Lösung herausgearbeitet. Hierdurch werden die Schülerinnen und Schüler für Gefahren innerhalb ihrer alltäglichen elektronischen Kommunikation sensibilisiert

Die Sicherheit kryptologischer Verfahren soll aufgrund eines vertieften Verständnisses beurteilt werden. Darüber hinaus sollen die Schülerinnen und Schüler mit der Anwendung kryptologischer Verfahren für die alltägliche Kommunikation vertraut gemacht werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Kommunikation in Rechnernetzen erklären

Dazu gehört:

- Strukturen von Kommunikationssystemen analysieren und beschreiben
- Kommunikation in Rechnernetzen erläutern und am Beispiel des Internet verdeutlichen
- Datensicherheit unter Berücksichtigung kryptologischer Verfahren erklären und beachten

Strukturen von Kommunikationssystemen analysieren und beschreiben	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Elemente von Kommunikationssystemen	– Einfache Kommunikationsvorgänge experimentell durchführen (z.B. Morsen, Lichtsignale, Handzeichen) und dabei die Konzepte „Sender“, „Empfänger“, „Nachricht“ und „Protokoll“ erarbeiten. – Protokolle als Summe aller Vereinbarungen zwischen Sender und Empfänger zur Abwicklung von Kommunikationsvorgängen beschreiben.
Eigenschaften von Kommunikationssystemen	– Anhand historischer Verfahren (z.B. optischer Telegrafie) und Kommunikationsvorgängen aus dem Alltag (z.B. Telefonieren, Briefversand) Eigenschaften von Kommunikationssystemen herausarbeiten. – Den Unterschied zwischen verbindungsloser und verbindungsorientierter bzw. bestätigter und unbestätigter Kommunikation verdeutlichen.

Kommunikation in Rechnernetzen erläutern und am Beispiel des Internet verdeutlichen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Grundprobleme und Lösungsansätze	– Grundprobleme unterschiedlicher Systeme erkennen und Lösungsansätze erarbeiten: – Bitübertragung zwischen zwei Rechnern (einfache serielle Übertragung von Datenpaketen) – Erkennung von Bitübertragungsfehlern (Prüfsumme, zyklische Redundanzüberprüfung CRC) – Reaktion auf Übertragungsfehler (automatische Wiederholungsanfrage, Quittungsbetrieb) – Datenkollision bei Systemen mit mehreren Sendern und Empfängern (weitere Informationen unter den Stichworten ALOHA, CSMA/CD) – Wegewahl in einem Netz von Netzen (Routingtabelle, Routing Information Protocol)
Schichtenarchitektur	➢ Anhand eines einfachen Schichtenmodells die Vorteile einer geschichteten System-Architektur besprechen ➢ Den Unterschied zwischen Diensten und Protokollen einer Schicht erläutern.
Dienste und Protokolle des Internet	➢ Kommunikationsvorgänge mit der Client-Server-Struktur beschreiben. ➢ Einen Überblick über aktuelle Dienste und Protokolle des Internet verschaffen. ➢ Protokolle (z.B. HTTP, POP, SMTP, FTP) mit Hilfe eines Terminalprogramms und mit Hilfe von RFC-Auszügen untersuchen. ➢ Mit Hilfe von vorhandenen Klassen, die eine TCP/IP-Verbindung herstellen (Sockets), eine Anwendung programmieren.

Datensicherheit unter Berücksichtigung kryptologischer Verfahren erklären und beachten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Sicherheitsziele	➢ Sicherheitsprobleme bei Kommunikationsvorgängen im Alltag aufzeigen. ➢ Die Brisanz von Sicherheitsproblemen bei elektronischer Kommunikation (z.B. Chat, E-Mail, Online-Banking, E-Vote) herausstellen und diskutieren. ➢ Vertraulichkeit, Authentizität, Integrität, Verbindlichkeit als Sicherheitsziele herausarbeiten.
Moderne Verfahren zur Verschlüsselung und Signierung	➢ Historische Verfahren nur als Anknüpfungspunkte an das Thema „Verschlüsselung“ nutzen. ➢ Das Grundprinzip asymmetrischer Verfahren erarbeiten. Dabei das Prinzip der Einwegfunktion besprechen und an Beispielen verdeutlichen. ➢ Das Grundprinzip des RSA-Verfahrens erarbeiten. ➢ Berechnungen von öffentlichen und geheimen Schlüsseln für einfache Zahlenbeispiele durchführen und Kodierungen bzw. Dekodierungen nachvollziehen. ➢ Die Sicherheit des RSA-Verfahrens erläutern und diskutieren. ➢ Mit aktuellen Werkzeugen ver-/ entschlüsseln und signieren (z.B. GnuPG).
Sicherheitsinfrastruktur	➢ Probleme zur Sicherheitsinfrastruktur besprechen: Schlüsselvergabe, Zertifizierung von Schlüsseln. ➢ Die Grundlagen einer Sicherheitsinfrastruktur durch Austausch und gegenseitige Signierung von Schlüsseln innerhalb des Kurses aufzeigen.

3.4 Algorithmen und Datenstrukturen

Algorithmen werden in der Informatik in vielen Bereichen zur Lösung von Problemstellungen benötigt - letztlich basiert jede Computeranwendung auf Algorithmen.

In der Sekundarstufe II sollen die im Wahlfach erworbenen Grundkenntnisse zu diesem Themenkomplex erweitert und vertieft werden. Dies führt zu einer systematischeren Beschäftigung mit Entwurf und Eigenschaften komplexerer Algorithmen und der zu Grunde liegenden Datenstrukturen.

Die Entwicklung von Algorithmen sollte stets von praktischen Problemstellungen ausgehen und in der Regel auch eine Implementierung in einer Programmiersprache einschließen. Eine eigenständige Behandlung von Grundstrukturen ist denkbar, der gesamte Themenbereich soll aber nicht als geschlossene Unterrichtsreihe behandelt, sondern mit anderen Bereichen vernetzt werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Algorithmische Problemlösungen entwickeln und bewerten

Dazu gehört:

- Die Bedeutung der algorithmischen Datenverarbeitung einschätzen
- Algorithmische Grundstrukturen beherrschen
- Algorithmen entwickeln und implementieren
- Algorithmen hinsichtlich Korrektheit und Effizienz überprüfen und bewerten

Die Bedeutung der algorithmischen Datenverarbeitung einschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Algorithmusbegriff	➢ Bewusst machen, dass jede automatisierte Verarbeitung von Daten mithilfe des Computers auf der Grundlage präziser Verarbeitungsvorschriften erfolgt. ➢ Den Algorithmusbegriff klären; Anforderungen an Algorithmen formulieren. ➢ Kriterien wie Endlichkeit, Ausführbarkeit, Eindeutigkeit und Allgemeinheit sollen herausgearbeitet werden.
Bedeutung von Algorithmen früher und heute	➢ Anhand von Beispielen aufzeigen, dass es Algorithmen schon gab, bevor es den Computer gegeben hat (z.B. Algorithmen in der Mathematik) ➢ Die besondere Bedeutung, die Algorithmen durch die Entwicklung des Computers gewonnen haben, bewusst machen (Möglichkeit der schnellen automatisierten Ausführung)

Algorithmische Grundstrukturen beherrschen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Basiskonzepte	➢ Die im Wahlfach bereits eingeführten Basiskonzepte (Variablenkonzept, Zuweisungskonzept, Typen) wiederholen und vertiefen (z.B. weitere Datentypen). ➢ Diese Basiskonzepte vielfach verwenden.
Kontrollstrukturen	➢ Die im Wahlfach bereits eingeführten Kontrollstrukturen mit ihren Darstellungsmöglichkeiten wiederholen und vertiefen (z.B Vergleich von Schleifentypen). ➢ Diese Kontrollstrukturen vielfach verwenden.

Rekursion	➢ Das Grundprinzip einer rekursiven Problemreduktion an einfachen Beispielen erarbeiten und dabei die Begriffe Rekursionsbasis und Rekursionsschritt zur Beschreibung von Problemreduktionen einführen. ➢ Die Relevanz rekursiver Algorithmusbeschreibungen durch komplexere Beispiele (z.B. Standardalgorithmen) verdeutlichen. ➢ Rekursive Problemlösungen in eher einfacheren Fällen selbstständig entwickeln. ➢ Vor- und Nachteile zwischen Rekursion und Iteration durch einen Vergleich herausstellen (u. a. Verständlichkeit, Effizienz).
Datenstrukturen	➢ Die bereits im Wahlfach eingeführte Datenstruktur Reihung wiederholen und zur Datenmodellierung verwenden. ➢ Weitere Datenstrukturen wie Liste, Baum problemorientiert thematisieren. ➢ Die Möglichkeiten der aktuellen Programmiersprache nutzen (z.B. eine vordefinierte Datenstruktur Liste). ➢ Komplexere Datenstrukturen mit Objekten modellieren.

Algorithmen entwickeln und implementieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Strategien zur Entwicklung algorithmischer Problemlösungen	➢ Problemsituationen analysieren (Anfangszustand, gewünschter Endzustand), anhand konkreter Beispiele verdeutlichen und differenziert beschreiben. ➢ Abläufe anhand konkreter Problemfälle durchspielen und die Ablaufmuster schrittweise mit Hilfe algorithmischer Konzepte beschreiben. ➢ Typische algorithmische Lösungsmuster besprechen (z. B. Zerlegung eines Problems in Teilprobleme, erschöpfende Suche, vollständige Fallunterscheidung). ➢ Mit Hilfe erlernter Strategien sowie eigener Ideen Problemlösungen selbstständig entwickeln.
Standardalgorithmen	➢ In verschiedenen Problemkontexten zentrale Standardalgorithmen der Informatik (z. B.: Such- und Sortieralgorithmen, Algorithmen zur Mustererkennung, schnelles Potenzieren, Verschlüsselungsalgorithmen, Routingalgorithmen, genetische Algorithmen) behandeln. ➢ Komplexere Algorithmen der Fachliteratur entnehmen und analysierend behandeln.

Prozedurkonzept	➢ Prozeduren und Funktionen als Mittel zur modularen Darstellung von Algorithmen einführen und vielfältig nutzen. ➢ Den Unterschied zwischen Prozeduren und Funktionen klären. ➢ Die Bedeutung von Parametern (formale, aktuelle) klären. ➢ Die Signatur einer Prozedur jeweils genau spezifizieren (Anzahl der Parameter, Typ der Parameter, ...)

Algorithmen hinsichtlich Korrektheit und Effizienz überprüfen und bewerten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Korrektheit	➢ Korrektheitsbetrachtungen von Beginn an durchführen. ➢ In einfachen Fällen die Korrektheit auch mit Hilfe von Invarianten zusichern. ➢ Das Verhalten des Algorithmus systematisch testen (z.B. Durchlaufen aller Pfade im Programmablauf). ➢ Fehler gegebenenfalls systematisch suchen (z.B. durch Einfügen zusätzlicher Ausgaben, schrittweises Abarbeiten im Debugger). ➢ Typische Fehlermuster besprechen (z. B. Bereichsüberschreitung bei einer Reihung).
Effizienz	➢ Die Bedeutung von Zeit- und Speicheraufwand bei der Entwicklung von Algorithmen klären. ➢ Exemplarisch Algorithmen hinsichtlich ihres Laufzeitverhaltens vergleichen. ➢ Den Zeitaufwand dabei experimentell oder durch Abschätzungen mit Hilfe einfacher Aufwandsmaße ermitteln.

3.5 Grenzen algorithmisch arbeitender Systeme

Eine wesentliche Erkenntnis der Informatik ist die Tatsache, dass Computern prinzipielle und praktische Grenzen gesetzt sind. Diese können auch in Zukunft, trotz ständig steigender Leistungsfähigkeit, nicht überschritten werden. Aufgrund ihrer Relevanz sollte diese Erkenntnis im Informatikunterricht angemessen thematisiert werden. Dabei ist eine Vernetzung mit Fragen nach den Grenzen der formalen Beschreibbarkeit von Sprachen mit Hilfe von Grammatiken und Automaten anzustreben.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Grenzen algorithmisch arbeitender Systeme einschätzen

Dazu gehört:

- Den Algorithmusbegriff präzisieren und das Präzisierungsverfahren bewerten
- Prinzipielle Grenzen algorithmisch arbeitender Systeme belegen und nachweisen
- Praktische Grenzen algorithmisch arbeitender Systeme aufzeigen

Den Algorithmusbegriff präzisieren und das Präzisierungsverfahren bewerten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Präzisierung des Algorithmusbegriffs	➢ Die Unzulänglichkeit einer informellen Begriffsbildung zur Klärung grundlegender Fragen nach der Tragweite der algorithmischen Problemlösemethode diskutieren. ➢ Die Turingmaschine als einfaches und historisch bedeutendes Berechnungsmodell zur Präzisierung einführen. ➢ Simulationsprogramme zur Veranschaulichung nutzen.
Berechenbarkeit und Entscheidbarkeit	➢ Die Begriffe „berechenbare Funktion“ und „entscheidbare Menge“ bzw. „entscheidbare Sprache“ präzisieren. ➢ In einfachen Fällen Berechenbarkeits- bzw. Entscheidbarkeitsnachweise führen, dabei auf einen extensiven „Programmierkurs“ im gewählten Präzisierungsansatz verzichten. ➢ Die Mächtigkeit des gewählten Berechnungsmodells „Turingmaschine“ aufzeigen, für komplexere Problemstellungen fertige Lösungen benutzen (z. B. beim Nachweis der Existenz einer universellen Turingmaschine).
Church-Turing-These	➢ Die Existenz weiterer Ansätze zur Präzisierung des Algorithmusbegriffs ansprechen. Zur Verdeutlichung eignen sich die Registermaschine als „hardware-naher“ Ansatz und die Programmiersprache WHILE als „software-naher“ Ansatz. ➢ Die Äquivalenz der verschiedenen Präzisierungsansätze plausibel machen. ➢ Die sich aus dem Äquivalenzsatz ergebende Church-Turing-These und ihre Relevanz zur Einordnung des intuitiven Algorithmusbegriffs diskutieren.

Prinzipielle Grenzen algorithmisch arbeitender Systeme belegen und nachweisen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
nicht berechenbare Funktionen	➢ Die Existenz nicht-berechenbarer Funktionen durch Abzählungsargumente belegen. ➢ Im Kontext „fleißige Biber“ die Rado-Funktion als Beispiel für eine nicht-berechenbare Funktion erleben.
nicht entscheidbare Mengen	➢ Das Halteproblem mit Hilfe (noch) nicht terminierender Programme verdeutlichen (z. B. im Kontext Primzahlen). ➢ Die algorithmische Unlösbarkeit des Halteproblems nachweisen. Wichtiger als formale Korrektheit ist dabei die Verdeutlichung der grundlegenden Beweisideen. ➢ Weitere Probleme, die zu nicht-entscheidbaren Mengen führen, thematisieren (z. B. 10. Hilbert’sches Problem, Grammatikprobleme).

Praktische Grenzen algorithmisch arbeitender Systeme aufzeigen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
exponentieller Aufwand	➢ Anhand geeigneter Probleme (z. B. Primfaktorzerlegung, Problem des Handlungsreisenden, Rucksackproblem) experimentell erfahren und durch Überlegungen herausfinden, dass es Probleme gibt, die zwar algorithmisch lösbar sind, deren Lösungsalgorithmen aber praktisch nicht genutzt werden können. ➢ Exponentiellen Aufwand mit Hilfe von Beispielrechnungen als Ursache für eine praktische Grenze erfahren. ➢ Den Unterschied zwischen polynomialem und exponentiellem Aufwand verdeutlichen.
Näherungslösungen	➢ An einem Beispiel aufzeigen, dass komplexe Probleme, für die es bisher nur praktisch undurchführbare algorithmische Lösungen gibt, in der Praxis mit Hilfe von Näherungsverfahren befriedigend gelöst werden (z. B. durch Einschränkung des Suchraums oder durch Simulation genetischer Verfahren).

3.6 Informatische Modellierung

Modellierung spielt innerhalb der Informatik eine zentrale Rolle. Informatiksysteme können aufgrund der zu bewältigenden Komplexität nicht ad hoc entwickelt werden. Man benötigt Baupläne, die den betreffenden Gegenstandsbereich sowie das gewünschte Systemverhalten vereinfachend und strukturgetreu beschreiben. Modellierung soll daher auch die Arbeitsweise im Unterricht prägen. Mit Hilfe von Modellen soll einerseits die Entwicklung von Problemlösungen systematisiert, andererseits das Verständnis von Informatiksystemen durch eine vereinfachende Darstellung unterstützt werden. Der Lehrplan sieht zwei Modellierungsansätze zur intensiveren Bearbeitung vor – zustandsbasierte und objektorientierte Modellierung.

Zustandsbasierte Modellierung erweist sich als einfacher und in vielen Bereichen der Informatik anwendbarer Modellierungsansatz, der im gesamten Informatikunterricht immer dann zur Analyse und Entwicklung von Systemen genutzt werden sollte, wenn das Verhalten eines Systems nicht nur von äußeren Eingaben oder Ereignissen abhängt, sondern auch von inneren Zuständen, die das System in Abhängigkeit von den Eingaben bzw. Ereignissen durchläuft.

Objektorientierte Konzepte nehmen in der Software-Entwicklung derzeit eine herausragende Stellung ein. Gängige Programmiersysteme bieten in der Regel umfangreiche Klassenbibliotheken an, mit deren Hilfe Objekte erzeugt werden können, die eine Vielzahl von Funktionalitäten bereit stellen. Im Informatikunterricht sollen die Grundideen und Grundlagen der Objektorientierung in didaktisch reduzierter Form behandelt werden, um vorgegebene Klassen nutzen zu können und die hiermit verbundenen Modularisierungsprinzipien kennen zu lernen.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Informatische Modelle entwickeln und implementieren

Dazu gehört:

- Zustandsbasierte Modelle zu komplexeren Problembereichen entwickeln
- Grundideen und Grundkonzepte der objektorientierten Modellierung erklären
- Objektorientierte Modelle zu komplexeren Problembereichen entwickeln und implementieren

Zustandsbasierte Modelle zu komplexeren Problembereichen entwickeln	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Basiskonzepte zustandsbasierter Modelle	➢ Anhand einfacher und aus dem Alltag bekannter Systeme die Basiskonzepte der zustandsbasierten Modellierung einführen: Zustand, Anfangszustand, Endzustände, Zustandsübergang.
Darstellung zustandsbasierter Modelle	➢ Zustandsgraph und Zustandstabelle zur Beschreibung zustandsbasierter Modelle einführen und nutzen.
Systembeschreibung mit zustandsbasierten Modelle	➢ Zustandsbasierte Modellierung vielfältig zur Beschreibung von einfachen und auch komplexeren Informatiksystemen nutzen. ➢ Ereignisgesteuerte Systeme als zustandsbasierte Systeme interpretieren (Ereignis als auslösende Aktion eines Zustandsübergangs; Ereignisbehandlung als hiermit verbundene ausgelöste Aktion)
Simulation zustandsbasierter Modelle	➢ Zustandsbasierte Modelle mit Hilfe von Werkzeugen simulieren. ➢ Programme zur Simulation zustandsbasierter Modelle erstellen und nutzen.

Grundideen und Grundkonzepte der objektorientierten Modellierung erklären	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Basiskonzepte der Objektorientierung: Klasse, Objekt, Nachricht	➢ Vorgegebene Klassenbibliotheken bei einer Problembearbeitung nutzen (z. B. zur Gestaltung einer Benutzungsoberfläche) und analysierend dabei erste Vorstellungen zum Objekt- und Klassenkonzept entwickeln: – Objekt als Einheit aus Attributen und Methoden, das für die Erledigung bestimmter Aufgaben zuständig ist – Objekt als aktive und autonome Programmeinheit mit eigener Identität, das durch eine Nachricht veranlasst werden kann, bestimmte Aufgaben zu erledigen – Klasse als Bauplan für Objekte, der für die Erzeugung von Objekten benötigt wird Schnittstelle der Klasse als Summe aller Informationen, die zur Benutzung der Klasse benötigt werden

	➢ Eigene Klassen zur Problembearbeitung entwickeln und dabei das Verständnis des Objekt- und Klassenkonzepts vertiefen: – Objekt als abstrahierende Entsprechung eines Gegen-stands aus dem Problembereich ➢ Klasse als Objekttyp, die Struktur und Verhalten der zu erzeugenden Objekte genau festlegt.
Basiskonzepte der Objektorientierung: Beziehungen	➢ Grundvorstellungen zum Beziehungskonzept entwickeln: – Beziehung zwischen Objekten als Grundlage dafür, dass mehrere Objekte gemeinsam eine komplexere Aufgabe erledigen – Herstellen von Beziehungen durch Referenzen. ➢ Verschiedene Beziehungsarten unterscheiden: Hat-Beziehung, Kennt- Beziehung, Ist- Beziehung ➢ Vererbung bei geeigneten Anwendungsbeispielen zur Verbesserung der Modellstruktur (z. B. Vermeidung von Code-Duplizierung) einsetzen. ➢ Vor- und Nachteile von Klassenhierarchien mit Vererbung diskutieren.
Prinzipien der Objektorientierung	➢ Die Relevanz objektorientierter Konzepte mit Prinzipien der Informatik verdeutlichen, dabei insbesondere folgende Aspekte diskutieren: – Klasse als abgeschlossene Einheit (Modularisierung) mit einer eindeutigen Schnittstelle, die verwendet werden kann, ohne die Details ihrer Realisierung zu kennen (Geheimnisprinzip) – Klasse als Programmierbaustein, der vielfältig verwendet werden und daher abstrahierend entworfen werden sollte (Abstraktion, Wiederverwendbarkeit).

Objektorientierte Modelle zu komplexeren Problembereichen entwickeln und implementieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Entwicklung objektorientierter Modelle	➢ Die Problembereiche zunächst einfach und anschaulich wählen, nach und nach den Komplexitätsgrad steigern. ➢ Funktionen eines Modells herausstellen und bei der Modellbildung beachten: – Modell als Abbild der Miniwelt – Modell als Vorlage für ein Informatiksystem. ➢ Modelle entsprechend schrittweise ausbauen: – Die Miniwelt analysieren und Klassen, Objekte mit ihren Zuständigkeiten und Beziehungen identifizieren – Die Aufgaben des geplanten Systems klären und bei der Weiterentwicklung von Modellen beachten.
Verwendung von Entwurfsmustern	➢ Elementare Entwurfsmuster herausstellen und bei der Modellierung nutzen, z.B. Trennung von Benutzungsoberfläche und dem Datenmodell (MVC - Konzept)
Darstellung objektorientierter Modelle	➢ Standardisierte Darstellungsformen (UML - Diagramme, wie z. B. Klassen-, Objekt- und Sequenz-Diagramme) einführen und zur Beschreibung von objektorientierten Modellen verwenden. ➢ Modelle mit Hilfe von Entwicklungswerkzeugen (z.B. speziellen Editoren) übersichtlich darstellen und dokumentieren.
Implementierung objektorientierter Modelle	➢ Implementierungsmuster für typische und häufig vorkommende Modellstrukturen erarbeiten und vielfältig nutzen, u. a. – Implementierung von Klassen – Erzeugung und Vernichtung von Objekten. ➢ Typische Fehler besprechen (z.B. Objekt wird benutzt, ist aber nicht erzeugt). ➢ Syntaxregeln der gewählten Programmiersprache thematisieren; Implementierungsdetails mit einem Hilfesystem nachschlagen ➢ Verfahren zur systematischen Fehlersuche erarbeiten (Fehler mit einem Debugger oder durch gezielt eingesetzte Ausgabeanweisungen lokalisieren).

3.7 Deklarative Programmierung

Der Lehrplan sieht als zentrales Programmierparadigma die objektorientierte Programmierung vor. Diese basiert in der Regel auf einem imperativen Kern, bei dem die anweisungsbasierte Modellierung von Abläufen das Denken prägt. Um die zu entwickelnden Denkschemata nicht einseitig imperativ auszurichten, soll der Unterricht auch einen Einblick in eine andere, deklarative „Programmierwelt“ gewähren. Dieser Einblick soll vor allem das andersartige Modellierungs- und Berechnungskonzept herausstellen und typischer Anwendungsfelder aufzeigen. Eine ausweitende Behandlung einer zweiten Programmierwelt einschließlich der dann erforderlichen programmiersprachlichen Details ist nicht beabsichtigt. Für den Unterricht stehen zwei Ansätze zur Auswahl, zum einen der logikbasierte Programmieransatz, bei dem der Problem-Lösungs-Zusammenhang mit Hilfe von Prädikatenlogik beschrieben wird, zum anderen der funktionale Programmieransatz, bei dem der Problem-Lösungs-Zusammenhang mit Hilfe von Funktionen beschrieben wird.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Deklarative Programmierung als alternativen Problemlöseansatz verwenden

Dazu gehört:

- Probleme mit logischer/funktionaler Programmierung lösen
- Die Relevanz logischer/funktionaler Programmierung einschätzen

Wahlpflichtthema I: Logische Programmierung

Probleme mit logischer Programmierung lösen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Modellierung mit Prädikatenlogik	– Das Modellierungskonzept der logischen Programmierung an Hand einfacher Beispiele erarbeiten: – Beschreibung der Objekte des Problembereichs mit Hilfe von Konstanten und Termen – Beschreibung von Objekt-Eigenschaften und Objekt-Beziehungen mit Hilfe von Prädikaten – Festlegung der Prädikate mit Hilfe von Fakten und Regeln – Nutzung von Rekursion zur Beschreibung komplexerer Zusammenhänge – Aufbau eines Programms aus einer Wissensbasis (Fakten und Regeln) und einer Anfrage an die Wissensbasis. ➢ Einfache Probleme mit Hilfe von Anfragen an eine logisch modellierte Wissensbasis lösen.
Berechnung durch logische Schlussfolgerungen	➢ Die Grundidee des logikbasierten Berechnungskonzepts herausarbeiten, um Einsicht in die Auswertung logischer Programme zu gewinnen: – Herleitung der Anfrageergebnisse aus den vorgegebenen Fakten und Regeln mit Hilfe der Schlussregel „modus ponens“ – Suche einer Herleitung mit Hilfe einer sog. Inferenzmaschine, die bei der Suche in dem komplexen Suchraum bestimmte Suchstrategien (Tiefensuche, Backtracking) verwendet ➢ Die Problematik, dass die Inferenzmaschine nicht immer das logisch intendierte Ergebnis liefert (z. B. bei Endlosschleifen) an Hand von Beispielen verdeutlichen

Die Relevanz logischer Programmierung einschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Imperative und logische Programmierung	– Den Unterschied zwischen imperativer und logischer Programmierung an Hand einfacher Beispiele herausarbeiten: – Imperative Programmierung: Steuerung einer konkreten oder abstrakten Maschine mit Hilfe von Anweisungen – Logische Programmierung: Beschreibung des Problembereichs mit Hilfe logischer Ausdrücke – Die Vor- und Nachteile der logischen Programmierung diskutieren: – Vorteile: kurze, direkt durchschaubare Programme – Nachteile: ggf. großer Berechnungsaufwand, Eignung für spezielle Problemklassen
Relevanz logischer Programmierung	➢ Die Relevanz der logischen Programmierung exemplarisch in einem typischen Anwendungsbereich aufzeigen – es eignen sich z. B. die Bereiche Datenbanken, Verarbeitung symbolischer Ausdrücke, Expertensysteme.

Wahlpflichtthema II: Funktionale Programmierung

Probleme mit Hilfe funktionaler Programmierung lösen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Modellierung mit Funktionen	– Das Modellierungskonzept der funktionalen Programmierung an Hand einfacher Beispiele erarbeiten: – Beschreibung der Objekte des Problembereichs mit Hilfe von Termen – Beschreibung der Abhängigkeiten zwischen den Objekten mit Hilfe von Funktionen – Festlegung der Funktionen mit Hilfe von Funktionsdeklarationen – Nutzung von Rekursion zur Beschreibung komplexerer Zusammenhänge – Aufbau eines funktionalen Programms aus einer Wissensbasis (endliche Menge von Funktionsdeklarationen) und einem Berechnungsterm (funktionaler Ausdruck). ➢ Einfache Probleme mit Hilfe funktionaler Programme lösen.
Berechnung durch Funktionsauswertung	➢ Die Grundidee des funktionalen Berechnungskonzepts herausarbeiten, um Einsicht in die maschinelle Auswertung funktionaler Programme zu gewinnen: – Herleitung der Berechnungsergebnisses durch Berechnung von Funktionswerten mit Hilfe der gegebenen Funktionsdeklarationen – Umsetzung mit Hilfe einer sog. Reduktionsmaschine, die den einfachsten Term durch wiederholte Funktionsanwendung (Reduktion) aus dem gegebenen Term erzeugt

Die Relevanz funktionaler Programmierung einschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Imperative und funktionale Programmierung	– Den Unterschied zwischen imperativer und funktionaler Programmierung an Hand einfacher Beispiele herausarbeiten: – Imperative Programmierung: Steuerung einer konkreten oder abstrakten Maschine mit Hilfe von Anweisungen – Funktionale Programmierung: Beschreibung des Problembereichs mit Hilfe funktionaler Ausdrücke – Die Vor- und Nachteile der funktionalen Programmierung diskutieren: – Vorteile: kurze, direkt durchschaubare Programme; keine Seiteneffekte bei der Berechnung der Ergebnisse – Nachteile: ggf. großer Berechnungsaufwand
Relevanz funktionaler Programmierung	➢ Die Relevanz der funktionaler Programmierung exemplarisch in einem typischen Anwendungsbereich aufzeigen – es eignen sich z. B. die Bereiche Verarbeitung symbolischer Ausdrücke, Entwurf von Interpretern.

3.8 Software-Entwicklung

Software-Entwicklung ist eine komplexe Aufgabe, an der üblicherweise mehrere Personen beteiligt sind und die zur Bewältigung gut geplantes Vorgehen erfordert. Im Rahmen von schulgerechten Software-Entwicklungsprojekten lassen sich erste Einblicke in Arbeitsprozesse gewinnen, die typisch für viele Bereiche in Wirtschaft und Technik sind. Schülerinnen und Schüler sollen hierbei u. a. erfahren, wie man komplexere Prozesse strukturiert und die Vorgehensweise organisiert, wie man im Team gemeinsam Teilaufgaben bearbeitet und dass man Verantwortung für das Gelingen des Projektes und für das entwickelte Produkt übernehmen muss. Bereits kleinere Entwicklungsprojekte zeigen zudem sehr schnell, wie schwierig es ist, zuverlässig funktionierende und benutzergerechte Software zu erstellen und helfen so, Anforderungen an Software-Qualität realistisch einzuschätzen.

Die Entwicklung kleinerer Software-Produkte kann zudem zum Verständnis vieler informatischer Konzepte beitragen. Indem man die Behandlung fundamentaler Konzepte mit passenden Anwendungsentwicklungen vernetzt, wird dieser Gegenstandsbereich durch die intensivere Beschäftigung vertieft.

Um Software-Entwicklungsprojekte erfolgreich zu gestalten, sollten die zu bewältigenden Aufgaben eher klein und überschaubar gehalten werden. Mehrere kleinere Projekte sind daher in der Regel einem einzigen größeren Projekt vorzuziehen. Hierdurch können auch die zur erfolgreichen Durchführung erforderlichen Kompetenzen gezielter weiterentwickelt werden.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Software verantwortungsbewusst, systematisch und kooperativ entwickeln.

Dazu gehört:

- Qualitätsmerkmale für Software kennen und beachten
- Software-Entwicklungsprozesse systematisch durchführen
- Ein Software-Entwicklungs-Projekt organisieren

Qualitätsmerkmale für Software kennen und beachten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Gütekriterien	➢ Anhand „berühmter Bugs" verdeutlichen, dass auch kommerzielle Software in der Regel Fehler enthält und sich gelegentlich unsicher verhält. ➢ Gütekriterien behandeln, die sich an DIN 66272 bzw. ISO/IEC 9126 orientieren: – *Funktionalität*: Besitzt die Software geforderten Funktionen? – *Zuverlässigkeit:* Ist die Software fehlerfrei und toleriert sie Fehleingaben? – *Benutzbarkeit:* Ist die Software leicht bedienbar? – *Effizienz:* Ist der Zeit- und Speicherbedarf angemessen? – *Änderbarkeit:* Kann die Software mit geringem Aufwand korrigiert oder erweitert werden? – *Übertragbarkeit:* Vermeidet die Software programmiersprachenspezifische Elemente? ➢ Immer wieder bei Software-Entwicklung einige dieser Kriterien gezielt thematisieren und bei der Realisierung beachten.
Verantwortlichkeit für Software	➢ Anhand von Fallbeispielen (z.B. Versagen medizinischer Software) die Verantwortlichkeit für Folgen von Software problematisieren. ➢ Ethik-Richtlinien diskutieren.

Software-Entwicklungsprozesse systematisch durchführen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Entwicklungsschritte: Anforderungsanalyse, Modellierung, Implementierung, Testen	➢ Die Vorgehensweise bei der Entwicklung von Software reflektieren und planen. ➢ Unabhängig von der Vorgehensweise folgende Schritte bei umfangreicheren Entwicklungsprozessen durchführen: – Die Anforderungen ermitteln (z. B. über Fallstudien) und mit einem Pflichtenheft dokumentieren – Modellierungsansätze passend auswählen (zustandsbasiert, objektorientiert, algorithmisch) und das zu entwickelnde System mit Hilfe von Modellen systematisch konzipieren

	– Die entwickelten Modelle mit der gewählten Programmiersprache implementieren – Alle Funktionalitäten des entwickelten Systems gezielt und isoliert testen ➢ Die Vorgehensweise der Lernsituation anpassen: Anfänger bedürfen einer stärkeren Unterstützung und Führung, während fortgeschrittene Lerner eigenständiger arbeiten sollten. ➢ Die Vorgehensweise der Problemsituation anpassen: Einfache Systeme können sequentiell (vom Problem direkt schrittweise zum Programm), komplexere dagegen iterativ und inkrementell (von ersten Prototypen über mehrere Zwischenprodukte bis hin zum voll funktionsfähigen System) entwickelt werden.
prozessbegleitende Dokumentation	➢ Alle Ergebnisse während der jeweiligen Schritte dokumentieren. ➢ Geeignete Werkzeuge zur Unterstützung einsetzen.

Ein Software-Entwicklungs-Projekt organisieren	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Organisationsbereiche: Projektauftrag, Aufgabenverteilung, Arbeitsplan, Absprachen	➢ Mehrere „kleinere“ Projekte durchführen, bei denen die Selbstständigkeit in Organisation und Durchführung zunehmen. ➢ Die Organisation von Software-Projekten vorbereiten: Organisationsaufgaben besprechen; zunächst sehr überschaubare Organisationsanlässe zum Einüben nutzen (z. B. ein gemeinsam entwickeltes, nicht sehr komplexes Modell in Teams arbeitsteilig implementieren, dabei auf sorgfältige Schnittstellenvereinbarungen achten). ➢ Aufgabenverteilung und Rollenzuweisung klar vereinbaren. ➢ Rahmenbedingungen vorab klären (insbesondere Zeitvorgaben). ➢ Kommunikations- und Kooperationssysteme zum Austausch von Informationen einsetzen. ➢ Sowohl Zwischen- als auch Endergebnisse präsentieren und diskutieren.

3.9 Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft

Die immer weiter fortschreitende Ausbreitung von Informationstechnik in alle Lebensbereiche macht es erforderlich, die Folgen dieser Entwicklung auf den Einzelnen und die Gesellschaft immer wieder zu reflektieren. Der Informatikunterricht ist diesem Anliegen in besonderer Weise verpflichtet, da hier fachliche Perspektiven stärker mitberücksichtigt werden können. Wechselwirkungen zwischen Informatiksystemen, Individuen und Gesellschaft treten in den verschiedensten unterrichtlichen Zusammenhängen auf und sollen im jeweiligen Sachkontext mitbehandelt werden. Die Inhalte zum Komplex Datensicherheit sind ebenfalls im Bereich Kommunikation in Rechnernetzen aufgeführt.

Vorrangiges Ziel des Unterrichts zu diesem Inhaltsbereich besteht im Erwerb der folgenden Kompetenz:

Mit Daten und Informatiksystemen verantwortungsvoll umgehen

Dazu gehört:

- Die Bedeutung der Informationstechnik für die Gesellschaft abschätzen
- Informationstechnik sozialverträglich gestalten und verantwortungsvoll einsetzen
- Datenerhebungen unter dem Aspekt Datenschutz beurteilen
- Kommunikation unter Aspekten der Datensicherheit bewerten
- Rechtliche Aspekte bei der Erstellung von Informatiksystemen berücksichtigen

Die Bedeutung der Informationstechnik für die Gesellschaft abschätzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Chancen und Risiken	➢ Die Bedeutung der Informationstechnik für die Gesellschaft anhand von z. T. „kritischen“ Einsatzbereichen (z.B. Kraftwerke, Flugzeuge, militärische und medizinische Anwendung) aufzeigen. ➢ Die Voraussetzungen (z.B. die Möglichkeit, auf Informationen zuzugreifen und sie zu verwenden) und Folgen (z.B. Ungleichheit in den individuellen Entwicklungsmöglichkeiten), Chancen (z. B. in der Medizin) und Risiken (z. B. durch Softwarefehler) bewusst machen und anhand aktueller Ereignisse diskutieren. ➢ Die Problematik von Sabotage durch Viren, Würmer und Trojaner deutlich machen.

Informationstechnik sozialverträglich gestalten und verantwortungsvoll einsetzen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Ethische Aspekte	➢ Die Frage nach der Verantwortung für Folgen der Informationstechnik besprechen und anhand von Fallbeispielen konkretisieren. ➢ Bei der Entwicklung eigener Software-Produkte die Entwicklung einer menschengerechten Informationstechnik erfahren und anschließend die Formen einer partizipativen Softwareentwicklung diskutieren. ➢ In diesem Rahmen auch auf Informatik bezogene Ethikkodizes (z.B. Ethische Leitlinien der Gesellschaft für Informatik) berücksichtigen.

Datenerhebungen unter dem Aspekt Datenschutz beurteilen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Missbrauch und Schutz personenbezogener Daten	➢ Das im Wahlfach behandelte Thema „Datenschutz“ erneut aufgreifen. Insbesondere auf die automatische Datenerhebung im Internet und bei der Internetnutzung eingehen. Hierbei auch die Interessen verschiedener Gruppen wie z.B. Nutzer, Provider, Wirtschaft und Staat erörtern und gegeneinander abwägen (eventuell in einem fächerverbindenden Projekt mit dem Fach Sozialkunde). ➢ Anhand von Beispielen besprechen, wie Daten erhoben bzw. gesammelt werden. Dabei in Fallstudien aufzeigen, dass bei einem sorglosen Umgang mit personenbezogenen Daten Missbrauch betrieben werden kann. ➢ Das Recht auf informationelle Selbstbestimmung besprechen. Anhand von Beispielen aufzeigen, dass das Recht auf Datenschutz unter bestimmten Umständen eingeschränkt ist. ➢ Das Verbot mit Erlaubnisvorbehalt, („alles was nicht erlaubt wird, ist verboten“) als Grundlage für die Datenschutz-Vorschriften besprechen.

Kommunikation unter Aspekten der Datensicherheit bewerten	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
Apekte der Datensicherheit	➢ Aspekte der Datensicherheit (Vertraulichkeit, Authentizität, Integrität, Verbindlichkeit) an alltäglichen Kommunikationssituationen (Briefe, Telefon,...) erarbeiten. ➢ Ihre Brisanz bei modernen Formen der elektronischen Kommunikation (Chat, E-Mail, Online - Banking, E - Vote,...) herausstellen und diskutieren.

Rechtliche Aspekte bei der Erstellung von Informatiksystemen berücksichtigen	
Verbindliche Inhalte	**Hinweise für eine mögliche Umsetzung**
einfache Rechtsgrundlagen	➢ Schülerinnen und Schüler für einen rechtlich einwandfreien Umgang mit Information sensibilisieren, die sich z.B. dann ergeben, wenn eigene Web-Seiten mit „fremdem“ Inhalten aufgefüllt werden. Hierzu gehört auch der Hinweis auf das Teledatenkommunikationsgesetz. ➢ Die Schülerinnen und Schüler mit Lizenzierungsmöglichkeiten z.B. GPL, Freeware, Shareware vertraut machen. ➢ Anhand von Fallstudien die Nutzungsrechte bei Software, Texten, Bildern, Musik, Filmen in Grundzügen besprechen. Dabei kann es notwendig sein, auch auf Fragen zum Patentrecht einzugehen. ➢ Anhand von Fallstudien auf die von Film- und Musikindustrie eingesetzten Kopierschutzprogramme eingehen. ➢ Die Schwierigkeiten ansprechen, die durch verschiedene Rechtssysteme entstehen (z.B. Veröffentlichung extremistischer Inhalte, Hinweis auf das aktuelle Strafrecht).