Ministerium für Bildung und Kultur
SAARLAND

Lehrplan

Informatik

Gymnasiale Oberstufe

Leistungskurs

Hauptphase

– Erprobungsphase –

2019

Inhalt

Vorwort

Seit der Entdeckung des Konzeptes der universellen Rechenmaschine und der Formalisierung des Algorithmenbegriffs in den 1930er Jahren hat die Informatik in rasanten Schritten den Alltag durchdrungen und gleichzeitig einen enormen Einfluss auf andere Wissenschaftsbereiche ausgeübt. Ein Schulfach, das dieser schnellen Entwicklung gerecht werden will, wird zwangsläufig Wert darauf legen müssen, Schülerinnen und Schülern eine Auswahl an Basiskonzepten zu vermitteln, die ihnen – in Anlehnung an den Begriff der *fundamentalen Idee* – eine breite, wenn auch exemplarische, Grundlage liefert.

Der vorliegende Lehrplan umfasst dementsprechend eine Auswahl an zentralen Themengebieten, die innerhalb der Informatik vielfältig vernetzt sind und auf diese Weise ihren Beitrag zur Beantwortung der wesentlichen Frage, was sich hinter dem Begriff *Informatik* letztendlich verbirgt, leisten. Ein Großteil der Themen umreißt fachliche Gebiete, die in ihrer konzeptionellen Kombination aus Schlichtheit und gleichzeitiger Mächtigkeit den vielfach zitierten „Test der Zeit“ überdauert haben und somit fundiert sicherstellen, dass der Informatikunterricht sich nicht auf kurzlebige Trends stützt.

Neben den zu vermittelnden inhaltlichen Kompetenzen trägt der Lehrplan auch den prozessbezogenen Kompetenzen Rechnung, die in den *Einheitlichen Prüfungsanforderungen* der Kultusministerkonferenz als *Erwerb und Strukturierung informatischer Kenntnisse, Kennen und Anwenden informatischer Methoden, Kommunizieren und Kooperieren sowie Anwenden informatischer Kenntnisse, Bewerten von Sachverhalten und Reflexion von Zusammenhängen* benannt werden und sich somit eng an die *Bildungsstandards Informatik für die Sekundarstufe II* der *Gesellschaft für Informatik (GI)* sowie an die Richtlinienempfehlungen *Computer Science Curricula* der *Association for Computing Machinery (ACM)* anlehnen.

Die in diesem Lehrplan angeführten fachlichen Kompetenzen dienen einer detaillierten Spezifizierung der angegebenen Fachinhalte; die hier verwendeten Operatoren stellen eine unmittelbare Verbindung zu den erforderlichen Prozesskompetenzen her.

Bei der angegebenen Reihenfolge der fachlichen Inhalte sowie ihrer Anteile an den zur Verfügung stehenden Unterrichtsstunden handelt es sich grobe Richtlinien: Bei Planung und Durchführung des Unterrichts ist auf eine enge und sinnvolle Verzahnung der Themen zu achten, die ihrerseits den Schülerinnen und Schülern ein klares Bild von der Vernetzung und Relevanz der Themen innerhalb der Fachwissenschaft Informatik vermittelt.

Zugelassene Programmiersprachen sind DELPHI, JAVA und PYTHON.

Zum Umgang mit dem Lehrplan

Der Lehrplan ist nach Themenfeldern gegliedert. Zu jedem Themenfeld werden in einem didaktischen Vorwort die Bedeutung der Thematik für die Schülerinnen und Schüler, die didaktische Konzeption und Besonderheiten wie zum Beispiel notwendige didaktische Reduktionen beschrieben.

In zwei Spalten werden die Inhalte des Themenfeldes und die daraus resultierenden verbindliche Kompetenzerwartungen bzw. Aktivitäten von Schülerinnen und Schülern, die zum Kompetenzerwerb beitragen, formuliert.

Die Kompetenzerwartungen bzw. Aktivitäten von Schülerinnen und Schülern sind bewusst detailliert beschrieben. Dies geschieht mit dem Ziel, die Intensität der Bearbeitung möglichst präzise festzulegen. So kann vermieden werden, dass Themenfelder entweder zu intensiv oder zu oberflächlich behandelt werden. Die detaillierte Beschreibung darf hierbei nicht als Stofffülle missverstanden werden. Der Lehrplan beschränkt sich vielmehr auf wesentliche Inhalte und Themen, die auch Bezugspunkte für schulische und schulübergreifende Leistungsüberprüfungen sind. Darüber hinaus lässt der Lehrplan Zeit für Vertiefungen, individuelle Schwertpunktsetzungen, fachübergreifende Bezüge und die Behandlung aktueller Themen.

Für die verbindlichen Themenfelder sind als Richtwerte jeweils Prozentanteile der insgesamt in der Hauptphase der gymnasialen Oberstufe zu Verfügung stehenden Unterrichtszeit angegeben.

Die zeitliche Abfolge der Inhalte innerhalb des jeweiligen Schulhalbjahres kann den Unterrichtsgegebenheiten angepasst werden.

Die Hinweise geben Anregungen inhaltlicher und methodischer Art.

Themenfelder Hauptphase der gymnasialen Oberstufe

Die angegebenen Anteile beziehen sich auf die in der Hauptphase insgesamt zur Verfügung stehende Unterrichtszeit und sind nicht zuletzt wegen der engen Verzahnung der Themen als ungefähre Richtwerte zu verstehen.

Themenfelder 1. Halbjahr der Hauptphase	**Informatik LK**
Objektorientierte Modellierung / Programmierung	15 %
Rekursion	10 %
Suchen und Sortieren	10 %

Themenfelder 2. Halbjahr der Hauptphase	**Informatik LK**
Dynamische Datenstrukturen: lineare Listen und binäre Bäume	10 %
Kryptologie	15 %
Komplexitätstheorie	5 %

Themenfelder 3. und 4. Halbjahr der Hauptphase	**Informatik LK**
Berechenbarkeit und ihre Grenzen	10 %
Formale Sprachen	15 %
Graphentheorie	10 %

Objektorientierte Modellierung/Programmierung	**Informatik Hauptphase LK**

Der objektorientierte Ansatz gehört heute in der Informatik zu den am weitesten verbreiteten Programmierparadigmen. Dabei wird in aller Regel dem sogenannten *Message-Passing-Modell* gefolgt, dessen Grundsatz darin besteht, dass der Ablauf eines Programms einer Kommunikation zwischen den erzeugten Objekten entspricht. Damit sind wesentliche Komponenten gegeben, die ein „objektorientierter" Ansatz vorweisen sollte: Objekte mit internem Zustand, der Zusammenfassung dieser Objekten in Klassen mit klassenweise spezialisiertem Verhalten sowie der Möglichkeit einer Vererbung zwischen Klassen, die ein polymorphes Verhalten der Objekte erlaubt.

Der objektorientierte Ansatz fördert in hohem Maße Modellierungs- und Modularisierungskompetenzen und verpflichtet Schülerinnen und Schüler durch die Notwendigkeit der Festlegung auf präzise definierte Schnittstellen zu intensiver Kommunikation.

Inhalte	**Kompetenzerwartungen**
• Objektorientierung und das *Message-Passing-Modell* • Klassen, Objekte, Attribute • Konstruktoren und Methoden • Vererbung und Polymorphie	Die Schülerinnen und Schüler • entwerfen *Message-Passing-Modelle* zu vorgegebenen Sachzusammenhängen, • implementieren Klassen mit Attributen primitiver und höherer Typen, • implementieren Konstruktoren und erzeugen Objekte mittels Konstruktoraufruf, • implementieren Methoden mit und ohne Rückgabewert und unterscheiden dabei bzgl. der Kommunikation zwischen *Auftrag* und *Anfrage*, • erkennen die Notwendigkeit/Möglichkeit der *Spezialisierung* bzw. der Erstellung eines einheitlichen *Obertyps* und entwerfen entsprechende Vererbungshierarchien, • implementieren polymorphe Methoden, • stellen einzelne Klassen sowie Vererbungsstrukturen in einer üblichen UML-Notation dar.

Hinweise

Dem *Message-Passing-Modell* liegt im Wesentlichen die Idee zugrunde, dass die erzeugten Objekte miteinander kommunizieren und dies bei der Ausführung des Programms die einzige Möglichkeit des Informationsflusses ist. Die oft mit dem sogenannten „Geheimnisprinzip“ in Zusammenhang gebrachte Datenkapselung innerhalb der Objekte bringt insofern mit sich, dass die durch das Objekt gekapselten Daten ausschließlich indirekt (über entsprechende Kommunikation) zugänglich sind. Ein unmittelbarer Zugriff widerspricht demnach der Kernidee des Message-Passing-Modells und verhindert darüber hinaus die Reorganisation der Interna der Klasse.

Bei der Behandlung der Vererbung sollte im Unterricht Wert auf die Betonung der Tatsache gelegt werden, dass jede Vererbungsstruktur dafür sorgt, dass Objekte einer Unterklasse in dem Sinne polymorph sind, dass sie auch die Typen aller Oberklassen instanzieren. Bezüglich der Methoden schlägt sich die Polymorphie dadurch nieder, dass Methoden gleichen Namens innerhalb einer Klassenhierarchie mehrfach definiert sein können, und das im Hintergrund liegende System dynamisch – auf Basis der Typen des angesprochenen Objekts – die entsprechend ‚richtige‘ Methode auswählt.

Die Konzepte der Mehrfachvererbung sowie der Multimethoden, die – außerhalb des Message-Passing-Modells – bezüglicher mehrerer Klassen spezialisiert werden, brauchen im Unterricht nicht behandelt zu werden. Gleiches gilt für den Themenkomplex der Metaobjektprotokolle.

Einzelne Klassen sowie Vererbungsschemata sind in einer üblichen UML-Notation darzustellen.

Rekursion	**Informatik Hauptphase LK**

Basierend auf der Idee der *rekursiven Funktionen*, die in den früher 1930er Jahren insbesondere durch die Arbeiten von Kurt Gödel maßgeblich dazu beitrug, den Algorithmenbegriff zu formalisieren, erhielt das Konzept der rekursiven Programmierung früh Einzug in das Standardrepertoire der Informatik, und ist seitdem aus dem Gebiet der Algorithmik nicht mehr wegzudenken.

Ziel des Informatikunterrichts ist es, dieses Konzept vorzustellen, und es unmittelbar als mächtiges und gleichzeitig elegantes Werkzeug zu etablieren. Eine intensive Anwendung findet die Rekursion schließlich im Themenbereich *Suchen und Sortieren* sowie in einer Vielzahl der Algorithmen, die auf induktiv definierten Datenstrukturen wie Listen und Bäumen operieren.

Inhalte	**Kompetenzerwartungen**
• Rekursion als Problemlösungsstrategie • lineare/baumförmige Rekursion und ihre Aufrufschemata • endständige Rekursion und Akkumulatoren • Vergleich äquivalenter rekursiver und iterativer Algorithmenvarianten • rekursive Definition der Fakultät, der Fibonacci-Zahlen und der ganzzahligen Potenz *(square-and-multiply)*	Die Schülerinnen und Schüler • benennen die Merkmale rekursiver Methoden und Definitionen, • skizzieren lineare und baumförmige Aufrufschemata, • erkennen und verwenden endständige Rekursion unter Verwendung von Akkumulatoren, • wandeln iterative/rekursive Methoden ineinander um, • benennen und implementieren exemplarisch rekursive Definitionen der Fakultät, der Fibonacci-Zahlen und der ganzzahligen Potenz, • beurteilen Vor- und Nachteile rekursiver Problemlösungen.

Hinweise

Aufgrund des thematischen Bezugs empfiehlt sich eine enge Vernetzung der Themen Rekursion und Suchen und Sortieren.

Bei den Laufzeituntersuchungen rekursiv formulierter Algorithmen kann auf formalmathematische Betrachtungen verzichtet werden; es genügt eine intuitive Vorstellung der Aufrufschemata.

Suchen und Sortieren	**Informatik Hauptphase LK**

Such- und Sortierverfahren auf linearen Reihungen gehören seit langem zu den Klassikern der Algorithmik. Sie haben einen engen Bezug zu vielen alltäglichen, praktischen Situationen und eignen sich in besonderer Weise, eigene Verfahrensideen zu entwerfen, diese zu formalisieren, zu implementieren und schließlich vergleichend zu analysieren. Bzgl. der Analyse bilden sie einen leicht zugänglichen Startpunkt für eine formale Laufzeitanalyse, die im Rahmen der Komplexitätstheorie vertieft und ausgeweitet werden kann und soll.

Inhalte	**Kompetenzerwartungen**
• sequentielle Suche • binäre Suche • einfache Sortierverfahren: *bubblesort, selectionsort, insertionsort* • höhere Sortierverfahren: *mergesort, quicksort*	Die Schülerinnen und Schüler • implementieren sequentielle und binäre Suche auf eindimensionalen Feldern, • implementieren die Sortierverfahren *bubblesort, selectionsort, insertionsort* auf eindimensionalen Feldern und vollziehen diese an Beispielen nach, • analysieren das Laufzeitverhalten der einfachen Sortierverfahren, • formulieren jeweils die Grundidee der höheren Sortierverfahren *mergesort* und *quicksort* und vollziehen diese an Beispielen nach, • analysieren das Laufzeitverhalten der höheren Sortierverfahren.

Hinweise

Behandelt werden in diesem Themenbereich ausschließlich Verfahren auf linearen Datenstrukturen mit direktem, indizierten Zugriff. Ein Ausweiten der genannten Verfahren auf Listenstrukturen bietet sich zwar an, ist jedoch nicht obligatorisch.

Während die einfachen Sortierverfahren zu implementieren sind, sollte bei den höheren, effizienteren Verfahren der Schwerpunkt auf die jeweilige Kernidee und hier besonders auf die Rolle der Rekursion gelegt werden; eine Implementierung dieser Verfahren wird nicht explizit verlangt.

Die Laufzeitanalysen können in diesem Themenblock zunächst noch intuitiv durchgeführt werden. Eine Präzisierung erfolgt im Themenblock K*omplexitätstheorie*.

Dynamische Datenstrukturen: lineare Listen und binäre Bäume	**Informatik Hauptphase LK**

Lineare Listen und binäre Bäume spielen als Datenstrukturen eine zentrale Rolle bei der praktischen Verwaltung und Bearbeitung von Daten. Beide Strukturen treten in vielfach spezialisierten Varianten auf und nehmen seit jeher eine ganz wesentliche und fundamentale Stellung in traditionsreichen Programmiersprachen wie *Lisp* oder *Prolog* ein. Im Unterricht des Leistungskurses sollen die Schülerinnen und Schüler mit den Grundvarianten dieser Strukturen vertraut gemacht werden. Dazu gehören grundlegende Definitionen und die Idee der strukturellen Induktion ebenso wie die Einführung in eine einfache Algorithmik auf eben diesen Strukturen. Ein verstärktes Augenmerk liegt auf dem Zusammenhang zwischen der theoretischen Fundierung einerseits und der praktischen Anwendung andererseits.

Inhalte	**Kompetenzerwartungen**
• Definition *linearer Listen* sowie der Begriffe *head, tail, Länge* • Algorithmik auf linearen Listen: Suchen, Löschen, Ersetzen, Einfügen • die Datenstruktur *Stack* mit den Operationen *isEmpty*, *top*, *push*, *pop* • die Datenstruktur *Queue* mit den Operationen *isEmpty*, *front*, *enqueue*, *dequeue* • Definition *binärer Bäume* sowie der Begriffe *Wurzel*, *linker/rechter Teilbaum*, *Knoten*, *Tiefe* • Algorithmik auf binären Bäumen: Suchen, Bestimmen der Tiefe und der Knotenanzahl • Suchbäume • Algorithmik auf Suchbäumen: Suchen, Einfügen, Löschen • Baumtraversierungen: *preorder-/inorder-/postorder*-Durchläufe • strukturelle Induktion	Die Schülerinnen und Schüler • definieren lineare Listen, binäre Bäume und Suchbäume, • erläutern die zugehörigen Grundbegriffe, • stellen Listen und Bäume graphisch dar, • erläutern die angegebenen Algorithmen auf den Datenstrukturen und führen diese exemplarisch aus, • verwenden rekursive Algorithmen auf Listen und Bäumen, • demonstrieren die unterschiedlichen Arten des Baumdurchlaufs, • verwenden die strukturelle Induktion als Beweismethode und wenden diese auf einfache Beispiel (wie etwa die Knotenanzahl eines Binärbaums in Abhängigkeit seiner Tiefe) an.

Hinweise

In der Praxis existiert eine Vielzahl verschiedener Definitionen sowohl für die linearen Listen als auch für die binären Bäume. Bewusst wird an dieser Stelle von einer Vereinheitlichung abgesehen; dennoch wird Wert auf eine größtmögliche Analogie zwischen Listen und Bäumen gelegt. In diesem Sinn soll – nicht zuletzt um eine elegante rekursive Algorithmik zu ermöglichen – der induktive Charakter beider Strukturen im Vordergrund stehen.

Das Prinzip der *strukturellen Induktion* besagt (bezogen auf binäre Bäume), dass eine Eigenschaft *P* für alle Binärbäume gilt, falls die folgenden beiden Bedingungen erfüllt sind:

- *P* gilt für den leeren Baum,
- gilt *P* für die beiden Teilbäume eines jeden (nicht-leeren) Binärbaums *B*, so gilt sie auch für *B*.

Dies bedeutet also, dass die betrachtete Eigenschaft bei der Konstruktion komplexer Strukturen aus weniger komplexen erhalten bleibt. Die enge Verwandtschaft mit der *vollständigen Induktion* auf der Menge der natürlichen Zahlen liegt somit auf der Hand und kann unmittelbar auf beliebige induktiv definierte Strukturen – etwa die durch eine Grammatik erzeugte Sprache – übertragen werden.

Komplexitätstheorie	**Informatik Hauptphase LK**

Die Komplexitätstheorie bildet in ihrer zentralen Stellung innerhalb der Informatik ein wesentliches Bindeglied zwischen praktischer Anwendung einerseits und theoretischen, innerinformatischen Untersuchungen andererseits: Während sich aus praktischer Sicht bei jedem implementierten Algorithmus quasi automatisch die Frage nach seiner Laufzeit und nach seinem Speicherplatzbedarf stellt, gehört etwa das *P = NP* -Problem nach wie vor zu den meistbeachteten ungelösten Problemen der theoretischen Informatik. Bahnbrechende Arbeiten von Thomas Cook und Richard Karp in der frühen Entwicklung der Komplexitätstheorie führten hier jeweils zu einem Turing-Award. Gleichzeitig hat die Frage nach der sogenannten *Programmlängenkomplexität* zur intensiven Entwicklung der *Algorithmischen Informationstheorie* geführt.

Inhalte	**Kompetenzerwartungen**
• der Begriff der *Problemgröße* • Laufzeit und die Begriffe *best-case*, *average-case*, *worst-case* • der funktionale Zusammenhang zwischen Problemgröße und Laufzeit • Definition der *Landau'schen O*-Notation • Algorithmenklassifizierung anhand der *O*-Notation • die Klassen *P* und *NP* und das *P = NP* -Problem	Die Schülerinnen und Schüler • erläutern den Begriff der Problemgröße und setzen ihn in einen funktionalen Zusammenhang zur Laufzeit, • verwenden die Begriffe *best-case*, *average-case*, *worst-case,* • nennen die Definition der *O*-Notation, • analysieren das Laufzeitverhalten von Algorithmen und notieren dieses in der *O*-Notation, • beschreiben die Bedeutung der Klassen *P* und *NP* und erläutern das *P = NP* -Problem.

Hinweise

Während der Begriff des *Problems* und somit auch der der *Problemklassen P* und *NP* in dieser Unterrichtssequenz nicht formal definiert, sondern nur intuitiv erfasst werden sollen, ist die *O*-Notation (in Anlehnung an die Kenntnisse der Schülerinnen und Schüler aus dem Mathematikunterricht) explizit formal zu definieren. Neben dieser Definition sind weniger formale (und der Kommunikation dienliche) Sprechweisen wie „*Bubblesort hat die Laufzeit O(n²)*" in diesem Zusammenhang ausdrücklich akzeptiert und erwünscht.

Die Klassen *P* bzw. *NP* werden als Klassen derjenigen Probleme, die in Polynomzeit gelöst, resp. deren Lösungen in Polynomzeit verifiziert werden können, verstanden und stellen somit einen engen Bezug zum Phänomen der ‚praktischen Berechenbarkeit' (im Gegensatz zum theoretischen Berechenbarkeitsbegriff) her. Nichtdeterminismus und ein „Raten" von Lösungen müssen an dieser Stelle nicht explizit thematisiert werden.

Die Klasse der NP-vollständigen Probleme kann sicherlich im Unterricht nicht formal thematisiert werden, sollte aber aufgrund ihrer fundamentalen Bedeutung im Zusammenhang mit der Klasse NP zumindest erwähnt werden.

Berechenbarkeit und ihre Grenzen	**Informatik Hauptphase LK**

Einen zentralen Aspekt bei der Geburt der modernen Informatik stellt der Versuch dar, den Algorithmen- bzw. Verfahrensbegriff präzise und formal zu erfassen. Die ersten diesbezüglichen Erfolge wurden in den 1930er Jahren von führenden Logikern wie *Gödel*, *Church*, *Post* und *Turing* erzielt. Ihre zunächst grundlegend verschiedenen Ansätze stellten sich schnell als äquivalent und gleich mächtig heraus, ein Umstand, der schließlich zur Formulierung der *Church'schen These* führte.

Ein besonders anschaulicher Ansatz, der dem imperativen Programmierparadigma sehr nahe steht, ist das Konzept der *Turingmaschine*, das in dieser Unterrichtssequenz die Grundlage der formalen Untersuchung der Grenzen der Berechenbarkeit darstellen soll, und das nach wie vor allgegenwärtig in allen Bereichen der theoretischen Informatik ist.

Inhalte	**Kompetenzerwartungen**
• Bestandteile und Aufbau einer Turingmaschine • Darstellung von natürlichen Zahlen und Wahrheitswerten • Turingmaschinenprogramme: Addition, Subtraktion, Verdopplung, Paritätsprüfung • die *busy-beaver*-Funktion Σ und ihre Nicht-Berechenbarkeit • das Halteproblem für Turingmaschinen • Universalität	Die Schülerinnen und Schüler • beschreiben den Aufbau einer Turingmaschine und benennen ihre Bestandteile, • entwerfen exemplarisch einfache Turingmaschinen-Programme, die auf natürlichen Zahlen und Wahrheitswerten operieren, • erläutern die busy-beaver-Funktion und skizzieren die Beweisidee der Nicht-Berechenbarkeit, • benennen das Halteproblem für Turingmaschinen und skizzieren einen zugehörigen Beweis, • erläutern den Begriff der *universellen Maschine*.

Hinweise

Betrachtet werden deterministische Turingmaschinen mit einem bidirektional unendlichen Band. Das Bandalphabet ist die Menge {`0`;`1`}. Dabei entspricht das Symbol `0` dem Wahrheitswert *false*, das Symbol `1` dem Wahrheitswert *true*. Die natürliche Zahl *n* wird durch eine zusammenhängende Sequenz der Länge *n* des Symbols `1` repräsentiert. Erhält ein Programm mehrere natürliche Zahlen als Eingabe, so stehen diese durch eine `0` getrennt auf dem Band. Zu Beginn der Ausführung steht der Kopf rechts unter der Eingabe, die (im Falle mehrerer Eingaben) am weitesten rechts steht. Nach Beendigung des Programms steht der Kopf rechts unter der Ausgabe. (Eingaben und Zwischenresultate müssen dabei nicht gelöscht werden.)

Turingmaschinenprogramme werden tabellarisch dargestellt. Dabei wird einem Paar aus aktuellem Zustand und gelesenem Bandsymbol ein Tripel aus zu schreibendem Symbol, Kopfbewegung und neuem Zustand zugeordnet. Kopfbewegungen werden dabei durch die Symbole `L` und `R` repräsentiert. Die Zuordnung muss nicht total sein.

Aufgrund der historischen Entwicklung bietet es sich in dieser Sequenz nicht an, die Turingmaschine zwangsläufig als einen erweiterten endlichen Automaten anzusehen. Nichts desto trotz sollte auf den engen Bezug aber hingewiesen werden.

Auf eine formale Definition der Turingmaschine kann verzichtet werden. Es gilt vielmehr, den Schülerinnen und Schülern eine klare anschauliche Vorstellung von der Arbeitsweise der Maschine zu vermitteln.

Kryptologie	**Informatik Hauptphase LK**

Mit der Möglichkeit der Kommunikation geht unmittelbar auch das Bedürfnis nach Vertraulichkeit einher. Vertrauliche Nachrichten sind nur für die vom Versender vorgesehenen Kommunikationspartner bestimmt, andere Personen sollen dabei keine Kenntnis der Nachrichten erhalten.

In der Geschichte finden sich zahleiche Beispiele von vertraulicher Kommunikation: Schon die Spartaner benutzten vor über 2500 Jahren Verfahren zur Geheimhaltung von Nachrichten. Über die römischen Feldherren bis hin zur ENIGMA-Maschine im 2. Weltkrieg sind historische Verfahren zur Geheimhaltung von Information bekannt. Eines haben alle relevanten historischen Verfahren gemeinsam: Die Motivation zur Geheimhaltung ging dabei vielfach von militärischen Interessen aus.

Erst seit der Digitalisierung der Kommunikation Ende des 20. Jahrhunderts rücken Verfahren zur Geheimhaltung stärker in den Fokus der Allgemeinheit. Große Datenmengen mit zum Teil sensiblen Komponenten werden von Privatpersonen, Behörden und Firmen digital versendet. Erstmals in der Geschichte können diese ungeschützten Daten – in den falschen Händen – gespeichert, verarbeitet und zusammengefügt werden, sofern sie nicht entsprechend geschützt werden.

Die Unterrichtsreihe zur Kryptologie greift die historischen Verfahren auf und klärt an diesen die benötigten Grundbegriffe. Es werden moderne Verfahren erarbeitet, die Geheimhaltung, Authentizität und Integrität gewährleisten. Die mathematische Grundlage bildet nahezu durchweg die modulare Arithmetik.

Inhalte	**Kompetenzerwartungen**
• Kryptographische Grundbegriffe: *Klartext, Geheimtext, Schlüssel, Prinzip von Kerckhoffs* • Sicherheit: *Vertraulichkeit*, *Authentizität*, *Integrität* • Chiffrieralgorithmen: *Substitutions*- und *Transpositionsalgorithmen* • klassische symmetrische Verschlüsselungsverfahren: *Skytale*, *Cäsar*, *Vigenère* • Angriffsarten: *Brute-Force*, *Known-Plaintext*, *Häufigkeitsanalysen* • Schlüsselaustauschverfahren: *Diffie-Hellman*, *Man-in-the-middle-Angriffe* • asymmetrische Verschlüsselung und digitale Signatur: Das *RSA-Verfahren* • die Idee *hybrider* Verschlüsselungsverfahren	Die Schülerinnen und Schüler • nennen und erläutern die angegebenen kryptographischen Grundbegriffe, • unterscheiden und erkennen *Substitutions*- und *Transpositionsalgorithmen,* • ver- und entschlüsseln vorgegebene Texte mit den Verfahren *Skytale*, *Cäsar*, *Vigenère*, • benennen die angegebenen Angriffsarten und führen sie für die Cäsar- und Vigenère-Verschlüsselung durch, • führen das Diffie-Hellman-Verfahren exemplarisch an kleinen Werten durch und erläutern seine Grundidee, • beschreiben den Man-in-the-middle-Angriff auf das Diffie-Hellman-Verfahren, • wenden den euklidischen und den erweiterten euklidischen Algorithmus an, • führen Schlüsselerzeugung, Ver- und Entschlüsselung des RSA-Verfahrens exemplarisch an kleinen Werten durch, • erläutern die Idee der digitalen Signatur anhand des RSA-Verfahrens, • erläutern die Idee *hybrider* Verschlüsselungsverfahren.

Hinweise

Das Schlüsselaustauschverfahren von Diffie und Hellman gibt nicht nur einen ersten Einblick in die moderne Kryptologie, sondern eignet sich insbesondere auch zur Überleitung in die asymmetrischen Verfahren: Hebt man die Symmetrie des Verfahrens auf und geht zu einer Veröffentlichung der zuerst berechneten Schlüsselkomponente über, so gelangt man automatisch zur Kernidee der asymmetrischen Verschlüsselung und vollzieht gleichzeitig die historische, gleich durch zwei Turing-Awards gekrönte Entwicklung nach. Überlegungen zur Man-in-the-middle-Attacke führen geradewegs zur Idee der digitalen Signatur.

Der Themenbereich „Hybride Verschlüsselungsverfahren“ fasst alle bisherigen Verfahren zusammen und bewertet sie hinsichtlich Vertraulichkeit, Authentizität und Integrität. Beispielhaft soll erläutert werden, dass letztendlich nur eine Kombination von Verfahren zum Schlüsselaustausch aber auch zur Verschlüsselung praktikabel umzusetzen sind. Geeignete Software kann den Schülerinnen und Schülern die praktische kryptologische Arbeit nahebringen.

Bei der Thematisierung der modernen Verfahren darf auf mathematische, den zugrunde liegenden Restklassenring betreffende Details und dementsprechend auf einen Nachweis der Korrektheit verzichtet werden. Nichtsdestotrotz können und sollen die Schülerinnen und Schüler aber etwa eine Anwendung der Potenzgesetze beim Diffie-Hellman-Verfahren erkennen und nachvollziehen können.

Formale Sprachen	**Informatik Hauptphase LK**
Einen einführenden Zugang zu den theoretischen Grundlagen der Informatik ermöglicht die Untersuchung endlicher Automaten und einfacher Klassen formaler Sprachen. Endliche Automaten werden einerseits als abstrakte Modelle zur Beschreibung von Abläufen und der Funktionsweise von Maschinen eingeführt, andererseits ermöglichen sie eine einfache und zugleich präzise Beschreibung regulärer Sprachen. Dieser Zusammenhang zwischen Zeichenmengen und sie erkennender Automaten führt zur Definition des Begriffs der formalen Sprache. Am Beispiel der regulären und kontextfreien Sprachen werden die praktische Bedeutung formaler Sprachen bei der Kommunikation zwischen Mensch und Maschine sowie ihre Beschreibungsmöglichkeiten erläutert.	
Inhalte	**Kompetenzerwartungen**
Formale Sprachen • Definition des Begriffs der *formalen Sprache* • Beschreibungen durch Aufzählung bzw. durch charakterisierende Eigenschaften der Worte Endliche Automaten mit und ohne Ausgabe • formale Definition deterministischer und nichtdeterministischer endlicher Automaten als 5-Tupel (Akzeptor) • Beschreibung der Funktionsweise durch Übergangs- und Ausgabetabellen sowie durch Übergangsgraphen • Konstruktion endlicher Automaten zu vorgegebenen Problemstellungen • Teilmengenkonstruktion Grammatiken • formale Definition einer Grammatik als 4-Tupel • Wortprüfung durch Ableitung • Chomsky-Hierarchie, Beispiele nicht-regulärer und nicht-kontextfreier Sprachen • Umwandlung regulärer Grammatiken und zugehöriger akzeptierender Automaten ineinander.	Die Schülerinnen und Schüler • definieren den Begriff der formalen Sprache und beschreiben formale Sprachen durch Aufzählung bzw. durch die Angabe charakterisierender Eigenschaften, • definieren akzeptierende endliche Automaten als 5-Tupel, • stellen endliche Automaten durch Übergangsgraphen sowie durch Übergangs- und Ausgabetabellen dar, • konstruieren endliche Automaten zu vorgegebenen Problemstellungen, • erstellen deterministische endliche Automaten mit Hilfe der Teilmengenkonstruktion, • definieren Grammatiken als 4-Tupel, • bestimmen die Ableitung vorgegebener ableitbarer Worte, • charakterisieren die vier Stufen der Chomsky-Hierarchie verbal und nennen Beispiele nicht-regulärer und nicht-kontextfreier Sprachen, • wandeln reguläre Grammatiken und entsprechende akzeptierende endliche Automaten ineinander um.

Hinweise

Endliche Automaten zur Modellierung von Abläufen und zur Spezifikation des Verhaltens einfacher Maschinen können an Beispielen aus der Praxis (Getränkeautomat, Parkscheinautomat) erklärt werden. Diese und weitere Beispiele aus der Steuertechnik führen zur Einführung der endlichen Automaten mit Ausgabe. Der endliche Akzeptor ist ein wichtiger Sonderfall endlicher Automaten. An den Beispielen eines Erzeugers für Paritätsbits und eines Prüfers für Paritätsbits kann die Funktionsweise der endlichen Automaten mit Ausgabe und der endlichen Akzeptoren verdeutlicht werden. Die Konstruktion endlicher Akzeptoren zu Problemstellungen wie der Suche vorgegebener Teilstrings in Zeichenketten (*pattern-matching*) führt zum Phänomen nichtdeterministischer Automaten. Endliche Automaten, die Folgen von Zeichen über einem Eingabealphabet akzeptieren, führen zum Begriff der formalen Sprache.

Im Gegensatz zur Menge der Worte einer natürlichen Sprache können viele formale Sprachen über die Beschreibung kennzeichnender Eigenschaften der Wörter oder über Bildungsregeln festgelegt werden. Für viele formale Sprachen ist eine vollständige Beschreibung durch charakterisierende Eigenschaften der Wörter nicht möglich. Grammatiken sind Regelwerke, welche die Herleitung aller Wörter einer Sprache ermöglichen.

Der Zusammenhang zwischen Grammatiken und Sprachen wird an einfachen Beispielen erarbeitet. Von wesentlicher Bedeutung ist hierbei die Klärung der (im Allgemeinen unentscheidbaren) Frage, ob eine vorgegebene Folge von Terminalsymbolen nach den Produktionsregeln einer vorgegebenen Grammatik aus der Startvariablen abgeleitet werden kann.

Graphentheorie	**Informatik Hauptphase LK**
Im ursprünglichen Sinn ist die Graphentheorie ein Teilgebiet der Diskreten Mathematik und beschäftigt sich mit den Eigenschaften von Graphen und ihren Beziehungen untereinander. Die Anwendungsvielfalt in der Informatik hat das Thema aber schließlich zu einem eigenständigen Bereich mit vielen Anwendungsfällen in zahlreichen Informatikgebieten werden lassen. Der Beginn lässt sich auf die Lösung des *Königsberger Brückenproblems* von *Leonard Euler* aus dem Jahr 1736 zurückführen. In seiner Publikation *Solutio problematis ad geometriam situs pertinentis* reduzierte er die Fragestellung auf ein topologisches Problem und begründete somit quasi den Anfang der Graphentheorie mit seiner Lösung: Es kommt in diesem Fall nicht auf die genaue Lage der Brücken an, sondern nur darauf, welche Brücke welche Inseln miteinander verbinden.	
Inhalte	**Kompetenzerwartungen**
• Definition gerichteter Graphen als Paar aus Knoten und Kanten • ungerichtete Graphen als Spezialfall gerichteter Graphen • Grundbegriffe: Knotengrad; schlichte, vollständige und zusammenhängende Graphen; gewichtete Graphen • Graphendarstellung: mengentheoretische Darstellung sowie Repräsentation durch Adjazenzlisten und Adjazenzmatrizen • Bäume als gerichtete, azyklische Graphen • Algorithmik auf Graphen: *Breiten-* und *Tiefensuche*, Konstruktion von *Spannbäumen*, das *Kürzeste-Wege-Problem* und der *Dijkstra-Algorithmus*, *Hamilton Kreise* und das *Traveling-Salesman-Problem*	Die Schülerinnen und Schüler • definieren Graphen als Paare aus einer Knoten- und einer Kantenmenge. • erläutern den Zusammenhang zwischen gerichteten, ungerichteten und gewichteten Graphen. • erläutern graphentheoretische Grundbegriffe: schlicht, vollständig, zusammenhängend. • stellen Graphen mengentheoretisch und mit Hilfe von Adjazenzlisten und Adjazenzmatrizen dar. • betrachten Bäume als spezielle Graphen. • modellieren informatische und Alltagssituationen als Graphen. • führen Breiten- und Tiefensuche auf gegebenen Graphen aus und erläutern die wesentlichen Unterschiede. • benennen das Kürzeste-Wege-Problem sowie das Traveling-Salesman-Problem und lösen diese algorithmisch. • führen einen Algorithmus zur Konstruktion von Spannbäumen durch.

Graphentheorie	**Informatik Hauptphase LK**

Hinweise

Die Graphentheorie soll sich als Teilgebiet der Diskreten Mathematik mit den Eigenschaften von Graphen und ihren Beziehungen untereinander beschäftigen. Es soll in diesem Thema Wert darauf gelegt werden, möglichst einfach und anschaulich die Zusammenhänge in der Graphentheorie zu erläutern. Die Schülerinnen und Schüler sollen in die Lage versetzt werden, einfache Probleme in eine Graphendarstellung zu überführen und entsprechende Algorithmen anwenden zu können. Sie kennen Anwendungen der Graphentheorie aus verschiedenen Gebieten der Informatik und können Graphenalgorithmen anwenden und ausführen.

In dieser Unterrichtseinheit werden Graphen als Paare einer Knotenmenge sowie einer Kantenmenge angesehen. Dabei ist die Kantenmenge wiederum eine Relation auf der Menge der Knoten. Viele der Eigenschaften eines Graphen können insbesondere an dieser Relation festgemacht werden. So wird etwa im Fall einer symmetrischen Relation von *ungerichteten* Graphen gesprochen. Im Falle von gewichteten Graphen werden die Elemente (Paare) der Kantenmenge durch ein sogenanntes *Gewicht*, in der Regel also eine reelle Zahl, auf Tripel erweitert.