Ernst-Moritz-Arndt-Gymnasium Bonn

Schulinterner Lehrplan

Informatik

Stand: 31.10.2019, VDB

1 Die Fachgruppe Informatik des EMA Bonn

Das Fach Informatik wird am Ernst-Moritz-Arndt-Gymnasium Bonn ab der Jahrgangsstufe 8 im Wahlpflichtbereich II unterrichtet. In diesem zweijährigen Kurs werden in altersstufengerechter Weise unter anderem die Grundlagen der Algorithmik am Beispiel verschiedener didaktischer Lernumgebungen (AppInventor, Robot Karol und Python) sowie die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen thematisiert werden. Darüber hinaus erhalten die Schülerinnen und Schüler dieses Differenzierungskurses einen tiefergehenden Einblick in Tabellenkalkulationen sowie die Gestaltung und Programmierung von Internetseiten (HTML, CSS).

Im Rahmen der Informations- und Kommunikationstechnologischen Grundbildung erhalten die Schülerinnen und Schüler des EMA in den Jahrgangsstufen 5, 6 und 7 einen Einblick in die grundlegende Funktionsweise von Rechner- und Dateisystemen, erlernen eine Textverarbeitung, erkennen und diskutieren Gefahren im Internet.

In den Klassen 6 und 7 wird darüber hinaus in Kooperation mit den Fächern Mathematik und Musik eine Einführung in eine Tabellenkalkulation und eine Präsentationssoftware gegeben.

Insofern verfügen alle Schülerinnen und Schüler unabhängig von der Fächerwahl in der Oberstufe über ein breit angelegtes Wissen zu den Grundlagen der Informations- und Kommunikationstechnologien, auf die im Unterricht der Sekundarstufe II aufgebaut werden kann.

Darüber hinaus haben die Schülerinnen und Schüler die Möglichkeit Informatik im Rahmen der individuellen Förderung in einer MINT-Leistungsgruppe in der Stufe 7 zu belegen.

Das Fach Informatik wird in der Sekundarstufe II am EMA Bonn als Wahlfach angeboten. In der Einführungsphase können die Schülerinnen und Schüler das Fach als dreistündigen Grundkurs belegen. In der Qualifikationsphase bietet das Gymnasium nun schon mehrere Jahre nacheinander einen Grund- und Leistungskurs im Fach Informatik an.

Um insbesondere Schülerinnen und Schülern, die in der Sekundarstufe I keinen Informatikunterricht besucht haben, gerecht zu werden, wird in den Kursen der Einführungsphase besonderer Wert darauf gelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Bestehen des Kurses erforderlich sind.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern und zu optimieren, entspricht der Informatikunterricht in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern ohne zu überfordern.

Darüber hinaus trägt er zu einer breitgefächerten Allgemeinbildung bei, bietet gleichzeitig Raum für individuelle Spezialisierungen und ermöglicht verantwortungsvolles Handeln in einer sich schnell wandelnden und von technischen Fortschritten geprägten Welt.

Die gemeinsame Entwicklung von Materialien und Unterrichtsvorhaben, die Evaluation von Lehr- und Lernprozessen sowie die stetige Überprüfung und eventuelle Modifikation des schulinternen Curriculums durch die Fachkonferenz Informatik stellt einen wichtigen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

Zurzeit besteht die Fachschaft Informatik des EMA Bonn aus fünf Lehrkräften mit Sek-II-Lehrbefähigung. Das EMA Bonn verfügt über zwei Computerräume mit 21 bzw. 16 Windows-Computerarbeitsplätzen und einer Bibliothek mit 6 Arbeitsplätzen. Zudem besitzt die Schule mittlerweile 16 digitale Tafeln, welche nicht nur im Informatikunterricht Verwendung finden. Alle Computerarbeitsplätze sowie die digitalen Tafeln sind am schulinternen Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule Zugriff auf ihre eigenen Daten zur Recherche im Internet oder Bearbeitung schulischer Aufgaben haben.

In den Kursen des Faches Informatik werden zwischen 15 und 25 Schülerinnen und Schüler unterrichtet. Leistungskurse sind in der Regel kleiner. Die Arbeit am Rechner erfolgt also im Normalfall in Einzel- oder Partnerarbeit.

- Fachvorsitzender: Nils van den Boom
- Stellvertreter: Benjamin Reichelt

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, sämtliche im Kernlehrplan angeführten Kompetenzen abzudecken. Dies entspricht der Verpflichtung jeder Lehrkraft, Schülerinnen und Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.

Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

Im "Übersichtsraster Unterrichtsvorhaben" (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnungen der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann. Um Spielraum für Vertiefungen, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z. B. Praktika, Kursfahrten o. ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans nur ca. 75 Prozent der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum "Übersichtsraster Unterrichtsvorhaben" zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, besitzt die exemplarische Ausweisung "konkretisierter Unterrichtsvorhaben" (Kapitel 2.1.2) empfehlenden Charakter. Referendarinnen und Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen, fachgruppeninternen Absprachen zu didaktisch-methodischen Zugängen, fachübergreifenden Kooperationen, Lernmitteln und Lernorten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.3 zu

entnehmen sind. Abweichungen von den vorgeschlagenen Vorgehensweisen bezüglich der konkretisierten Unterrichtsvorhaben sind im Rahmen der pädagogischen Freiheit der Lehrkräfte jederzeit mögliche. Sicherzustellen bleibt allerdings auch hier, dass im Rahmen der Umsetzung der Unterrichtsvorhaben insgesamt alle konkretisierten Kompetenzerwartungen des Kernlehrplans Berücksichtigung finden.

Da in den folgenden Unterrichtsvorhaben Inhalte in der Regel anhand von Problemstellungen in Anwendungskontexten bearbeitet werden, werden in einigen Unterrichtsvorhaben jeweils mehrere Inhaltsfelder angesprochen.

2.1.1 Übersichtsraster Unterrichtsvorhaben

I) Einführungsphase – Grundkurs

Im Folgenden sollen Unterrichtsvorhaben für das Fach Informatik dargestellt werden. Alle hier aufgeführten Vorhaben beziehen sich auf Grundkurse in der Einführungsphase.

Zu jedem Unterrichtvorhaben ist eine Anknüpfung an den Kernlehrplan Informatik in Form von Kompetenzbezügen gegeben. Die aufgeführten Kompetenzen sind dabei so zu verstehen, dass das entsprechende Unterrichtsvorhaben zum Erwerb derselben beiträgt. Kompetenzerwerb ist ein kontinuierlicher und kumulativer Prozess, der sich über längere Zeiträume hinzieht und die wiederholte Beschäftigung mit entsprechenden fachlichen Gegenständen und Themen in variierenden Anwendungssituationen oder auf zunehmenden Anforderungsniveaus voraussetzt. Es kann daher nicht der Anspruch erhoben werden, dass die aufgeführten Kompetenzen nach Abschluss lediglich eines Unterrichtsvorhabens vollständig erworben wurden.

Übersichtsraster Unterrichtsvorhaben: Einführungsphase – Grundkurs:

Unterrichtsvorhaben E-I Thema: Überblick über das Fach Informatik, Grundlagen von Informatiksystem und Geschichte der Informatik Zentrale Kompetenzen: Argumentieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Digitalisierung Einzelrechner Dateisysteme Internet Einsatz von Informatiksystemen Wirkungen der Automatisierung Geschichte der automatischen Datenverarbeitung Zeitbedarf: 6 Stunden	Unterrichtsvorhaben E-II Thema: Grundlagen der Programmierung mit Java und einfache Algorithmik Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Informatiksysteme Inhaltliche Schwerpunkte: Analyse, Entwurf und Implementierung einfacher Algorithmen Syntax und Semantik einer Programmiersprache Zeitbedarf: 33 Stunden
Unterrichtsvorhaben E-III Thema: Entwurf und Implementierung einfacher Sortierverfahren und eines klassischen kryptographischen Verfahrens Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Inhaltliche Schwerpunkte: Analyse, Entwurf und Implementierung einfacher Algorithmen Syntax und Semantik einer Programmiersprache Algorithmen zum Suchen und Sortieren Zeitbedarf: 12 Stunden	Unterrichtsvorhaben E-IV Thema: Objektorientierte Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen, Simulationen und grafischen Oberflächen Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung einfacher Algorithmen Syntax und Semantik einer Programmiersprache Zeitbedarf: 36 Stunden

II) Qualifikationsphase – Grundkurs/Leistungskurs

Qualifikationsphase I	
Unterrichtsvorhaben Q1-I Thema: Rekursive Algorithmen und Backtracking in Anwendungskontexten Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Algorithmen Formale Sprachen und Automaten Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in ausgewählten informatischen Kontexten Syntax und Semantik einer Programmiersprache Nutzung von Informatiksystemen Grenzen der Automatisierung Zeitbedarf: 20/25 Stunden	Unterrichtsvorhaben Q1-II Thema: Modellierung und Implementierung dynamischer Listenstrukturen und deren Anwendungen Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Informatiksysteme Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in ausgewählten informatischen Kontexten Syntax und Semantik einer Programmiersprache Nutzung von Informatiksystemen Zeitbedarf: 25/30 Stunden
Unterrichtsvorhaben Q1-III Thema: Modellierung und Implementierung dynamische nichtlineare Datenstrukturen am Beispiel der Binärbäume Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Informatiksysteme	Unterrichtsvorhaben Q1-IV Thema: Sicherheit und Datenschutz in Netzstrukturen und Modellierung und Implementierung von Client-Server-Anwendungen Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten

Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in ausgewählten informatischen Kontexten Syntax und Semantik einer Programmiersprache Nutzung von Informatiksystemen Zeitbedarf: 20/25 Stunden	Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in ausgewählten informatischen Kontexten Syntax und Semantik einer Programmiersprache Einzelrechner und Rechnernetzwerke Sicherheit Nutzung von Informatiksystemen, Wirkungen der Automatisierung Zeitbedarf: 15/25 Stunden
Summe Qualifikationsphase I: 80 (105)	

Qualifikationsphase II	
Unterrichtsvorhaben Q2-I / Q1-V Thema: Modellierung und Nutzung relationaler Datenbanken in Anwendungskontexten Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprache und Automaten Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Datenbanken Algorithmen in ausgewählten informatischen Kontexten Syntax und Semantik einer Programmiersprache Nutzung von Informatiksystemen Sicherheit Wirkung der Automatisierung	Unterrichtsvorhaben Q2-II Thema: Endliche Automaten und Formale Sprachen Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Informatiksysteme Inhaltliche Schwerpunkte: Endliche Automaten (und Kellerautomaten) Grammatiken regulärer (und kontextfreier) Sprachen Scanner, Parser und Interpreter für eine reguläre Sprache Möglichkeiten und Grenzen von Automaten und formalen Sprachen Nutzung von Informatiksystemen Zeitbedarf: 25/40 Stunden

Zeitbedarf: 25/30 Stunden	
Unterrichtsvorhaben Q2-III Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit Zentrale Kompetenzen: Argumentieren Kommunizieren und Kooperieren Inhaltsfelder: Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Einzelrechner und Rechnernetzwerke Grenzen der Automatisierung Zeitbedarf: 10 Stunden	Q2-IV Thema: Modellierung und Implementierung dynamischer nichtlinearer Datenstrukturen am Beispiel der Graphen Zentrale Kompetenzen: Argumentieren Modellieren Implementieren Darstellen und Interpretieren Kommunizieren und Kooperieren Inhaltsfelder: Daten und ihre Strukturierung Algorithmen Formale Sprachen und Automaten Informatiksysteme Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in ausgewählten informatischen Kontexten Zeitbedarf: 15 Stunden
Summe Qualifikationsphase II: 60 (95)	

2.1.2 Konkretisierte Unterrichtsvorhaben

Hinweis: Thema, Inhaltsfelder, inhaltliche Schwerpunkte, Kompetenzen und Absprachen zur vorhabenbezogenen Konkretisierung hat die Fachkonferenz verbindlich vereinbart. In allen anderen Bereichen (Unterrichtssequenzen und verwendeten Beispiele, Medien und Materialien) sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich. Darüber hinaus enthält dieser schulinterne Lehrplan in den Kapiteln 2.2 bis 2.3 übergreifende sowie z. T. auch jahrgangsbezogene Absprachen zur fachmethodischen und fachdidaktischen Arbeit, zur Leistungsbewertung und zur Leistungsrückmeldung. Je nach internem Steuerungsbedarf können solche Absprachen auch vorhabenbezogen vorgenommen werden.

I) Einführungsphase – Grundkurs

Im Folgenden sollen die in Abschnitt 1 aufgeführten Unterrichtsvorhaben konkretisiert werden. Diese Konkretisierung hat vorschlagenden Charakter, ohne die pädagogische Freiheit des Lehrenden einschränken zu wollen.

Die übergeordneten Kompetenzen des Kompetenzbereichs "Kommunizieren und Kooperieren" werden in jedem Unterrichtsvorhaben erworben bzw. vertieft und sind daher nicht jedes Mal erneut aufgeführt.

Kommunizieren und Kooperieren (K)

Schülerinnen und Schüler

- verwenden Fachausdrücke bei der Kommunikation über informatische Sachverhalte,
- kommunizieren und kooperieren in Gruppen und in Partnerarbeit,
- präsentieren Arbeitsabläufe und Arbeitsergebnisse.

Unterrichtsvorhaben Nr. 1

Thema: Überblick über das Fach Informatik, Grundlagen von Informatiksystem und Geschichte der Informatik

Leitfragen: Mit welchen Themen befasst sich das Fach Informatik in der Schule? Wie funktioniert ein moderner Computer? Welche Entwicklung durchlief die moderne Datenverarbeitung?

Zeitbedarf: 9 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Da einige Schülerinnen und Schüler das Fach zum ersten Mal in der Einführungsphase belegen, wird zu Beginn ein Überblick über die Themen des Schulfachs Informatik gegeben. Unter anderem wird auf den zentralen Begriff der Information eingegangen und die Möglichkeit der Codierung von Daten, insbesondere wird die Binärdarstellung von Zahlen thematisiert. Stationen der geschichtlichen Entwicklung werden angesprochen wie z.B. prinzipieller Prozessoraufbau, von-Neumann-Architektur und das EVA-Prinzip. Außerdem werden die Schülerinnen und Schüler in die konkrete Nutzung der Informatiksysteme an der Schule eingewiesen, insbesondere gehört dazu die Nutzung einer Lernplattform.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Allgemeine Einführung a) Übersicht über das Fach b) Einführung in die Informatiksysteme der Schule, insbesondere Lernplattform	Die Schülerinnen und Schüler • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D), • nutzen das verfügbare Informatiksystem zu strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	Lernplattform moodle

2. Grundlagen von Informatiksystemen und Überblick über die Geschichte der Informatik a) Darstellung von Zahlen im Binärsystem b) Von-Neumann-Architektur c) Geschichte der Datenverarbeitung	Die Schülerinnen und Schüler • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D), • beschreiben und erläutern den strukturellen Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A). • erläutern wesentliche Grundlagen der Geschichte der digitalen Daten-verarbeitung (A).	

Unterrichtsvorhaben Nr. 2

Thema: Grundlagen der Programmierung mit Java und einfache Algorithmik

Leitfragen: Aus welchen Bausteinen besteht ein Java-Programm? Welche Datentypen und Kontrollstrukturen stehen in Java zur Verfügung und wie nutzt man diese? Wie verläuft der Entwicklungsprozess eines Java-Programms?

Zeitbedarf: 33 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Anhand des Aachener Leitprogramms „Einführung in die Programmierung mit Java“ werden die Grundlagen zur Programmiertechnik mit Java durch die Schülerinnen und Schüler selbstständig und individuell erarbeitet. Dazu zählen: einfache Datentypen, Variablenkonzept, Wertzuweisungen, Kontrollstrukturen, Methodenkonzept, Arrays. Die Schülerinnen und Schüler wenden ihr Wissen kontextbezogen auf kleinere Problemstellungen an. Dabei durchlaufen sie den kontinuierlichen Prozess von Implementieren, Compilieren und Testen.

Die Schülerinnen und Schüler dokumentieren kontinuierlich und selbstständig ihren Lernfortschritt in einem Lerntagebuch.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Einführung in die Programmierung a) Kap. 1: Erste Schritte b) Kap. 2: Programmaufbau c) Kap. 3: Datentypen und Operationen	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A),	Leitprogramm „Einführung in die Programmierung“ Java-Editor Lerntagebuch

d) Kap. 4: Variablen e) Kap. 5: Verzweigungen f) Kap. 6: Schleifen g) Kap. 7: Arrays h) Kap. 8: Methoden	• modifizieren einfache Algorithmen und Programme (I), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • testen Programme schrittweise anhand von Beispielen (I). • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	

Unterrichtsvorhaben Nr. 3

Thema: Entwurf und Implementierung einfacher Sortierverfahren und eines klassischen kryptologischen Verfahrens

Leitfragen: Wie kann man die bisherigen Erkenntnisse im Umgang mit Arrays zum Sortieren von großen Datenmengen und zur einfachen „Verschlüsselung“ von Daten benutzen? Wie effizient ist ein Algorithmus hinsichtlich seines Laufzeitverhaltens?

Zeitbedarf: 12 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Aufbauend auf den programmiertechnischen Grundlagen des letzten Unterrichtsvorhabens können die Schülerinnen und Schüler nun ihr Wissen zu Arrays und Strings kontextbezogen anwenden:

Dazu werden mindestens zwei der einfachen Sortierverfahren behandelt, indem sie simuliert, implementiert und getestet werden. In der anschließenden Analyse reflektieren die Schülerinnen und Schüler das Laufzeitverhalten der Algorithmen im best-, worst- und average-case.

Darüber hinaus wird ein einfaches kryptologisches Verfahren (z.B. CAESAR) behandelt. Neben der Durchführung der einfachen Verschlüsselungstechnik werden die zur Implementierung nötigen String-Operationen vertieft. Hierzu gehört auch die Thematisierung des ASCII-Codes. Die Schülerinnen und Schüler reflektieren die Sicherheitsaspekte des Verfahrens.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Implementierung und Analyse einfacher Sortier- und Suchverfahren auf Arrays	Die Schülerinnen und Schüler • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D), • entwerfen einen weiteren Algorithmus zum Sortieren (M), • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeitaufwand und Speicherplatzbedarf (A).	Moodle MathePrisma Uni Wuppertal Youtube Java-Editor
2. Implementierung und Analyse eines einfachen kryptologischen Verfahrens	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • testen Programme schrittweise anhand von Beispielen (I).	Moodle MathePrisma Uni Wuppertal Java-Editor

Unterrichtsvorhaben Nr. 4

Thema: Objektorientierte Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen, Simulationen und grafischen Oberflächen

Leitfragen: Was sind Objekte, was sind Klassen? Wie programmiert man objektorientiert? Was bedeutet Vererbung? Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche realisieren? Wie lassen sich komplexere Datenflüsse und Beziehungen zwischen Objekten und Klassen realisieren?

Zeitbedarf: 36 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend von einem vorgegebenen Projekt (z. B. crazy kara) wird eine der Entwicklungsumgebungen BlueJ, Greenfoot, die 3D-Bibliothek GLOOP oder eine ähnliches Tool eingeführt. Diese wird genutzt, um den Schülerinnen und Schüler die Bedeutung von Objekten und Klassen deutlich zu machen. Innerhalb des Projekts modellieren sie eigene Klassen und differenzieren diese in ihren Attributen und Methoden. Neben der Erzeugung von Objekten erlenen sie auch deren Benutzung und Interaktionen untereinander.

Die Klassen des Projektes werden schrittweise durch Vererbung konkretisiert und deren Funktion implementiert. Dabei werden die bereits erlernten Grunddatentypen und verschiedenen Arten von Methoden in anderem Zusammenhang angewendet und eingeübt. Insbesondere wird das Geheimnisprinzip vertieft. Zur Verdeutlichung der Referenzsemantik wird auf das „Faden-Referenzmodell“ zurückgegriffen.

Erlerntes wird in einer komplett selbst erstellten Animation oder einem komplett selbst erstellten Spiel reorganisiert. Das Projekt wird dabei geeignet dokumentiert.

Nach Abschluss des Projektes wird die objektorientierte Programmierung anhand des Entwurfs und der Implementierung grafischer Oberflächen mit dem Java-Editor kontextbezogen vertieft. Dabei werden die Ereignisbehandlungsroutinen der Oberflächen-Objekte verwendet und das Geheimnisprinzip angewendet.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Grundlagen der Entwicklungsumgebung und Einführung in die OO a) Was ist ein Objekt? Was ist eine Klasse? b) Objekte erzeugen c) Darstellung / Verhalten von Objekten d) Methoden und Attribute e) Objektinteraktion f) Vererbung g) Geheimnisprinzip h) Referenzsemantik	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assozia-tionsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von	Moodle Greenfoot BlueJ GLOOP Faden-Referenzmodell Fortbildungsmaterialien zur OO

	Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	
2. Erstellung einer eigenen Animation/eines eigenen Spiels a) Entwicklung einer Spiel-/Animationsidee (Anforderungsdefinition) b) Implementierung der Idee unter Verwendung von Hilfestellungen aus dem Internet c) Dokumentation des Projekts	zusätzlich: Die Schülerinnen und Schüler • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K),	Moodle Greenfoot GLOOP Fortbildungsmaterialien zur OO

	• nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	

II)/III) Qualifikationsphase – Grundkurs/Leistungskurs

Die Aufstellung der Unterrichtsvorhaben erfolgt für den Grundkurs und den Leistungskurs kombiniert.

Die folgenden Kompetenzen aus dem Bereich *Kommunizieren und Kooperieren* werden in allen Unterrichtsvorhaben der Qualifikationsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung /Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Ebenso bieten fast alle Unterrichtsvorhaben, in denen Programme implementiert werden, die Gelegenheit, die folgenden Kompetenzen zu erwerben bzw. zu vertiefen:

Schülerinnen und Schüler

- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- wenden eine didaktisch orientierte Entwicklungsumgebung (didaktisch orientierte Entwicklungsumgebungen) zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M).

Unterrichtsvorhaben Q1-I

Thema: Rekursive Algorithmen und Backtracking in Anwendungskontexten

Leitfragen: Wie können komplexe, rekursiv definierte Probleme informatisch gelöst werden? Gibt es schnelle (rekursiv definierte) Sortier- und Suchverfahren?

Zeitbedarf: 20/30 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend vom einem Problem wie z. B. "Türme von Hanoi" wird Rekursion als fundamentale Idee der Informatik zunächst im mathematischen, danach aber auch im informatischen Zusammenhang angewendet. Dabei wird zwischen linearen und verzweigten Rekursionen unterschieden und das Laufzeitverhalten bei hoher Rekursionstiefe analysiert.

Verschiedene NP-vollständige Probleme (wie z. B. Rucksack, n-Damen, Springer, Irrgarten, etc.) werden algorithmisch rekursiv formuliert und als Backtracking-Algorithmus implementiert.

Bereits bekannte Such- und Sortierverfahren (z. B. Sortieren durch Einfügen, Sortieren durch Auswahl, Sequentielle Suche) werden rekursiv formuliert und durch leistungsfähigere Verfahren (z. B. Quicksort, Mergesort, Heapsort, Binäre Suche) ergänzt. Die neuen Verfahren werden implementiert.

Lernmittel / Materialien:

- Arbeitsblätter in der moodle-Plattform
- eine didaktische Entwicklungsumgebung (z. B. Java-Editor o. a.)

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Entwicklung der Rekursion als fundamentale Idee der Informatik • rekursive Formeln • rekursive Funktionen / Methoden • rekursive Programmierung	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I),	Türme von Hanoi mit Schwerpunkt auf • Zahl der Versetzungsoperationen • Protokollierung der Versetzungen
2. Rekursion in mathematischen und informatischen Kontexten • Rekursion in mathematischen Kontexten • Analyse und Darstellung des rekursiven Ablaufs einer Methode • Analyse des Laufzeitverhaltens linearer und verzweigter Rekursion	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	• Fakultätsfunktion (lineare Rekursion) • Fibonacci-Funktion (verzweigte Rekursion) • ggT (verzweigte Rekursion) • evtl. Fraktale (Kochkurve, Sierpinskidreieck, etc.)
3. NP-vollständige Probleme lösen mit Backtracking • Erarbeitung verschiedener NP-vollständiger Probleme • Algorithmische Beschreibung einer Lösungsidee • Implementierung eines Problems als Backtrackingalgorithmus	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und "Backtracking" (M) • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	• Rucksackproblem • Irrgartenproblem • Projektarbeit zu einer Problemstellung

4. Effiziente Sortierverfahren / Suchverfahren • Wiederholung bereits bekannter Sortier- und Suchverfahren als rekursiver Algorithmus • Erarbeitung eines Sortierverfahrens der Laufzeit O(n*log(n)) • Erarbeitung eines Suchverfahrens der Laufzeit O(log(n)).	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und "Backtracking" (M), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	• Demonstrationsprogramm zur Visualisierung von Sortierverfahren • Quicksortvisualisierung zur Erarbeitung der Idee • Suchspiel zur Erarbeitung der Binären Suche

Unterrichtsvorhaben Q1-II

Thema: Modellierung und Implementierung dynamischer Listenstrukturen und deren Anwendungen

Leitfragen: Wie können beliebig viele linear angeordnete Daten im Anwendungskontext verwaltet werden?

Zeitbedarf: 25/30 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Stapeln am Beispiel dargestellt und die Operationen der Klasse Stack anhand der Abiturklasse erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert.

Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben. Anhand einer Anwendung, in der Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Dabei werden die Operationen der Datenstruktur Schlange thematisiert und die entsprechende Abiturklasse Queue verwendet.

Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse List gemäß der Abiturklasse eingeführt und in einem Anwendungskontext verwendet.

(Je nach thematisierten Anwendungskontexten ist eine andere Reihenfolge bei der Behandlung der oben genannten Datenstrukturen möglich und zulässig.)

In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Lernmittel / Materialien:

- Arbeitsblätter in der moodle-Plattform
- eine didaktische Entwicklungsumgebung (z. B. Java-Editor o. a.)
- Zeigermodell
- Modellprogramm zur Visualisierung dynamischer Listenstrukturen

<table>
<tr><th>Unterrichtssequenzen</th><th>zu entwickelnde Kompetenzen</th><th>Beispiele, Medien, Materialien</th></tr>
<tr><td>1. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack
• Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen
• Erarbeitung der Funktionalität der Klasse Stack
• Modellierung und Implementierung der Anwendung unter Verwendung der Klasse Stack.
• Implementierung der Klasse Stack</td><td rowspan="3">Die Schülerinnen und Schüler
• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),
• modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),
• ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
• stellen die Kommunikation zwischen Objekten grafisch dar (D),
• stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
• dokumentieren Klassen (D),
• analysieren und erläutern objektorientierte Modellierungen (A),
• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).
• analysieren und erläutern Algorithmen und Programme (A),
• modifizieren Algorithmen und Programme (I),
• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D),
• implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),</td><td>Visualisierungsprogramm zu dynamischen Datenstrukturen</td></tr>
<tr><td>2. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue
• Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen
• Erarbeitung der Funktionalität der Klasse Queue
• Modellierung und Implementierung der Anwendung unter Verwendung der Klasse Queue.
• Implementierung der Klasse Queue</td><td>Visualisierungsprogramm zu dynamischen Datenstrukturen</td></tr>
<tr><td>3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List
• Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen
• Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.
• Implementierung der Klasse List</td><td>Visualisierungsprogramm zu dynamischen Datenstrukturen
Einfache Oberfläche zur Visualisierung der linearen Liste, z. B. die Verwaltung einer Namensliste</td></tr>
</table>

	• testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • erläutern Operationen dynamischer (linearer oder/und nicht-linearer) Datenstrukturen (A), • implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I),	
4. Vertiefung / Anwendung einer linearen Datenstruktur im Anwendungskontext.	zusätzlich: Die Schülerinnen und Schüler • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),	Umsetzung in einem größeren Projekt

Unterrichtsvorhaben Q1-III

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen am Beispiel der Binärbäume

Leitfragen: Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden? Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden? Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?

Zeitbedarf: 20/25 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt.

Anschließend werden für eine Problemstellung in einem der Anwendungskontexte Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse BinaryTree der Vorgaben für das Zentralabitur NRW verwendet. Klassen und ihre Beziehungen werden in Entwurfs- und Implementationsdiagrammen dargestellt. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) implementiert werden. Unterschiede bezüglich der Möglichkeit, den Baum anhand der Ausgabe der Bauminhalte via Pre-, In- oder Postorder-Traversierung zu rekonstruieren, werden dabei ebenfalls angesprochen, indem die fehlende Umkehrbarkeit der Zuordnung Binärbaum => Inorder-Ausgabe an einem Beispiel verdeutlicht wird.

Eine Tiefensuche wird verwendet, um einen in der Baumstruktur gespeicherten Inhalt zu suchen.

Zu einer Problemstellung in einem entsprechenden Anwendungskontext werden die Operationen der Datenstruktur Suchbaum thematisiert und unter der Verwendung der Klasse BinarySearchTree der Vorgaben für das Zentralabitur weitere Klassen oder Methoden in diesem Kontext modelliert und implementiert. Die Suchbäume werden wie zuvor auch grafisch dargestellt.

Die Verwendung von binären Bäumen und Suchbäumen wird anhand weiterer Problemstellungen oder anderer Kontexten weiter geübt.

Lernmittel / Materialien:

- eine didaktische Entwicklungsumgebung (z. B. Java-Editor o. a.)
- Arbeitsblätter und Demonstrationsprogramme in der moodle-Plattform
- Modellprogramm zur Visualisierung dynamischer Listenstrukturen

<table>
<tr><th>Unterrichtssequenzen</th><th>zu entwickelnde Kompetenzen</th><th>Beispiele, Medien, Materialien</th></tr>
<tr><td>1. Analyse von Baumstrukturen in verschiedenen Kontexten
• Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit)
• Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten</td><td rowspan="3">Die Schülerinnen und Schüler
• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),
• modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
• modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie/oder lineare und nichtlineare Datensammlungen zu (M),
• verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),
• ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
• stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
• dokumentieren Klassen (D),
• analysieren und erläutern objektorientierte Modellierungen (A),
• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).
• analysieren und erläutern Algorithmen und Programme (A),</td><td>• Demoprogramm und Arbeitsblätter zum Projekt Akinator</td></tr>
<tr><td>2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree
• Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext
• Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms
• Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen
• Implementierung der Anwendung oder von Teilen der Anwendung
• Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf
• Implementierung der Standardoperationen der Klasse BinaryTree</td><td>• Arbeitsblätter zur Projektarbeit Akinator
• Traversierungsverfahren
• Projektarbeit Morsebaum</td></tr>
<tr><td>3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree
• Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen
• Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm,
• grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften</td><td>• Arbeitsblätter zum binären Suchbaum
• Arbeitsblätter zum Baumsortieren und zu Traversierungsverfahren</td></tr>
</table>

• Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation • Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums • Implementierung ausgewählter Standardoperationen der Klasse BinarySearchTree	• modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • erläutern Operationen dynamischer (linearer oder/und nicht-linearer) Datenstrukturen (A), • implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I),	
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		• Projektarbeiten zu einem der Themen Termbäume, Morsebäume, Stichwortbaum, Ahnenbaum

Unterrichtsvorhaben Q1-IV

Thema: Sicherheit und Datenschutz in Netzstrukturen und Modellierung und Implementierung von Client-Server-Anwendungen

Leitfragen: Wie werden Daten in Netzwerken übermittelt? Was sollte man in Bezug auf die Sicherheit beachten?

Zeitbedarf: 15/25 Stunden (Das Thema bietet Vertiefungen darüber hinaus an vielen Stellen an. Eine erste Erprobung des Unterrichtsvorhabens kann über Möglichkeiten hierzu Aufschluss geben und den zeitlichen Bedarf ggf. verändern.)

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend von einer Kommunikation zwischen zwei Kommunikationspartnern über eine einfache Leitung wird die Notwendigkeiten einer Datenübertragung erarbeitet. Die Schichten des TCP/IP-Schichtenmodells werden beispielgebunden erarbeitet (Basisbandübertragungsverfahren, Prüfverfahren, Vermittlungsschicht, Anwendungsprotokoll) und an einer Simulationssoftware getestet. Verschiedene Netzwerk-Topologien werden entwickelt und in Client-Server-Anwendungen simuliert.

Im Leistungskurs werden ausgehend von einer einfachen Echo-Anwendung die beteiligten Komponenten (Echoserver und Echoclient) entwickelt und unter Verwendung der ZA-Klassen implementiert.

Die Echo-Anwendung wird zu einer Chat-Anwendung erweitert, notwendige Protokolle werden entwickelt und systematisch dargestellt.

Die Schülerinnen und Schüler entwickeln eine individuelle Client-Serveranwendung, definieren notwendige Protokolle und erweitern die Chat-Anwendung entsprechend der Vorgaben.

Über die Sicherheit von Netzwerkanwendungen wird das Augenmerk auf verschiedene symmetrische und asymmetrische kryptografische Verfahren gelenkt, welche analysiert und erläutert werden. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.

Lernmittel / Materialien:

- Arbeitsblätter zur Einführung in Netzwerke
- Simulationsprogramm "Filius"
- Arbeitsblätter und Skript zu "Filius"
- Arbeitsblätter zum Chat-Projekt
- Kryptografie-Programm "Cryptool"
- Arbeitsblätter zu kryptografischen Verfahren

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Schichten des TCP/IP-Protokolls • Erarbeitung der Notwendigkeiten einer Netzwerkkommunikation • Erarbeitung der Schichten des TCP/IP-Protokolls: Ethernet-, Internet-, Transport- und Anwendungsschicht	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),	• Arbeitsblätter zur Einführung in Netzwerke
2. Simulation von Netzwerken / Netzwerk-Topologien • Erarbeitung der Topologien: Peer-to-Peer, Sterntopologie, Baumtopologie, Vermaschtes Netz • Simulation von Client-Server-Anwendungen • Simulation von Protokollen der Anwendungsschicht (POP3, SMTP, etc.)	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A),	• Simulationssoftware FILIUS • Arbeitsblätter und Skript zu FILIUS
3. Entwicklung verschiedener Client-Server-Anwendungen • Grundlegende Begriffe • ZA-Klassen • Echo-Anwendung • Chat Anwendung (Nutzung der List) • Beliebige Client-Server-Anwendung	• erläutern das Prinzip der Nebenläufigkeit (A), • analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), • entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I). • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M).	Umsetzung eines größeren Projektes • Chat • Anwendungsserver (wie z.B. Buchungsserver für Kinokarten)
4. Analyse und Erläuterung kryptografischer Verfahren • Erläuterung symmetrischer Verfahren: monoalphabetisch: Cäsar, polyalphabetisch: Vigenère • Erläuterung asymmetrischer Verfahren: RSA, Diffie-Hellman • Analyse der Sicherheit verschiedener Verfahren und Auswirkungen auf den Datenschutz/Urheberrecht	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A).	• Arbeitsblätter zu kryptografischen Verfahren • CrypTool • Materialien von klicksafe (Zusatzmodule „Nicht alles was geht, ist auch erlaubt“, „Ich bin öffentlich ganz privat“, „Datenschutz im Internet“)

	• untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).	

Unterrichtsvorhaben Q2-I

Hinweis: Im Leistungskurs sollte dieses Thema in die Q1 vorgezogen werden.

Thema: Modellierung und Nutzung relationaler Datenbanken in Anwendungskontexten

Leitfragen: Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden?
Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?

Zeitbedarf: 25/30 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend von einer konkreten Anwendungssituation entwickeln die Schülerinnen und Schüler Ideen zur Modellierung von Daten und erkennen die Vorzüge von Datenbanksystemen.

In weiteren Anwendungskontexten müssen Datenbanken entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schülerinnen und Schüler in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in das Relationale Modell überführt.

An einem Beispiel wird verdeutlicht, dass in Datenbanken Redundanzen unerwünscht sind und Konsistenz gewährleistet sein sollte. Die 1. bis 3. Normalform wird als Gütekriterium für Datenbankentwürfe eingeführt. Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Ausgehend von einer vorhandenen Datenbasis, für die eine Datenbank angelegt und mit den Daten gefüllt wird, entwickeln Schülerinnen und Schüler für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schülerinnen und Schülern analysiert und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden. Die Operationen der Relationenalgebra werden mit SQL-Abfragen simuliert.

Anhand von Fallbeispielen werden Probleme bei der Nutzung von Datenbanksystemen aufgezeigt und im Hinblick auf gesellschaftliche Auswirkungen diskutiert.

(Die exakte Abfolge der Teilvorhaben sowie die Verzahnung theoretischer und praktischer Anteile kann hierbei je nach Anwendungskontext flexibel angepasst werden.)

Lernmittel / Materialien:

- Arbeitsblätter und Demonstrationsprogramme in der moodle-Plattform
- Datenbankserver (z. B. XAMPP)
- SQL-Tutorial (z. B. die Seite SQL-Zoo, VideoCenter-Datenbank)

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Modellierung von relationalen Datenbanken • Entity-Relationship-Diagramm: Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms Erläuterung und Modifizierung einer Datenbankmodellierung • Entwicklung einer Datenbank aus einem Datenbankentwurf: Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln • Redundanz, Konsistenz und Normalformen: Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation. Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	Die Schülerinnen und Schüler • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M).	• Arbeitsblätter zur Einführung in Datenbanken in der moodle-Plattform
2. Nutzung von relationalen Datenbanken • Aufbau von Datenbanken und Grundbegriffe: Entwicklung von Fragestellungen zur vorhandenen Datenbank Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema	Die Schülerinnen und Schüler • modifizieren eine Datenbankmodellierung (M), • bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D),	• Arbeitsblätter zur Einführung in SQL in der moodle-Plattform • Tutorial der Seite SQL-Zoo (http://sqlzoo.net/wiki/Main_Page) • VideoCenter-Datenbank

• SQL-Abfragen: Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL) • Vertiefung an einem weiteren Datenbankbeispiel: Vertiefungen am Beispiel der Relationenalgebra	• analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I). • erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A),	
3. Gesellschaftliche Auswirkungen der Nutzung von Datenbanksystemen	• untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).	• Fallbeispiele zur Nutzung von Datenbanksystemen • Spiel zum Missbrauch von Daten: DataDealer (http://datadealer.com/de)

Unterrichtsvorhaben Q2-II

Thema: Endliche Automaten und Formale Sprachen

Leitfragen: Wie kann man endliche Automaten/Kellerautomaten genau beschreiben? Wie können endliche Automaten/Kellerautomaten modelliert werden? Wie können Sprachen durch Grammatiken beschrieben werden? Welche Zusammenhänge gibt es zwischen formalen Sprachen, Automaten und Grammatiken?

Zeitbedarf: 25/40 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend von einem konkreten Anwendungsbeispiel entwickeln die Schülerinnen und Schüler das Modell der Grammatik einer formalen Sprache und das Modell des endlichen Automaten. Die Schülerinnen und Schüler überführen Automaten in verschiedene Darstellungsformen und ermitteln die akzeptierte Sprache eines Automaten (z. B. in Form von regulären Ausdrücken). An einem Beispiel wird ein nichtdeterministischer Akzeptor als Alternative gegenüber einem entsprechenden deterministischen Akzeptor eingeführt.

Der Zusammenhang zwischen endlichen Automaten und regulären Grammatiken wird durch die Entwicklung allgemeingültiger Verfahren zur Transformation zwischen Automat und Grammatik dargestellt. Die Unzulänglichkeit endlicher Automaten und regulärer Grammatiken wird an Beispielen verdeutlicht.

Das Grammatikmodell der regulären Grammatiken wird auf das Modell der kontextfreien Grammatiken erweitert und die Auswirkungen auf das entsprechende Automatenmodell der Kellerautomaten veranschaulicht. Die Unzulänglichkeit der Kellerautomaten und kontextfreien Grammatiken wird an Beispielen verdeutlicht.

Ausgehend von einer einfachen formalen Sprache (z. B. reduziertes Java) werden die Bestandteile eines Compilers dargestellt:

Der Scanner eines Compilers wird in Form eines endlichen Automaten modelliert und implementiert. Die Begriffe der Symboltabelle und der Tokenliste werden inhaltlich gefüllt. Der Sprachumfang der einfachen formalen Sprache wird leicht erweitert und die Auswirkungen auf den Automaten und die Implementierung wird beobachtet.

Der Parser eines Compilers wird in Form einer kontextfreien Grammatik modelliert und implementiert. Der Sprachumfang der einfachen formalen Sprache wird um weitere Regeln ergänzt und der Parser wird angepasst.

Bei Bedarf wird ein Übersetzer-Modul entwickelt, welches die einfache formale Sprache in eine Sprachebene übersetzt (z. B. interpretiert und compiliert).

Lernmittel / Materialien:

- Arbeitsblätter und Demonstrationsprogramme in der moodle-Plattform
- Automatensimulationsprogramm (z. B. JFlap)

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Einführung in Automaten/Grammatiken • Grammatiken: Grammatik einer natürlichen Sprache Grammatik einer künstlichen Sprache Idee des Parsens • Automaten: erkennender Automat zu Symbolen einer Sprache Modell des endlichen Automaten Darstellungsformen Sprache eines Automaten als regulärer Ausdruck nichtdeterministische Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M).	Mögliche Einstiege: • Grundidee eines Compilers: Scanner, Parser, Codierer
2. Zusammenhang zwischen endlichen Automaten und regulären Grammatiken • reguläre Grammatik: Definition Anwendungen • Zusammenhang zu endlichen Automaten • Grenzen der endlichen Automaten/regulären Grammatiken	Die Schülerinnen und Schüler • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • entwickeln zur akzeptierten Sprache eines Automaten	Arbeitsblätter in der moodle-Plattform Orientierungsrahmen: Chomsky

	eine zugehörige Grammatik (M), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D), • zeigen/erläutern die Grenzen endlicher Automaten und regulärer Grammatiken/Sprachen im Anwendungszusammenhang auf (A).	
3. Zusammenhang zwischen Kellerautomaten und kontextfreien Grammatiken • kontextfreie Grammatik: Definition Anwendungen • Modell des Kellerautomaten Definition Darstellungsfomen Anwendungen / Sprache eines Kellerautomaten • Zusammenhang zwischen Kellerautomaten/kontextfreien Grammatiken • Grenzen der Kellerautomaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • beschreiben an Beispielen den Zusammenhang	Arbeitsblätter in der moodle-Plattform Orientierungsrahmen: Chomsky

	zwischen Automaten und Grammatiken (D).	
4. Die Schritte eines Compilers • Scanner: endlicher Automat als Grundlage Symboltabelle und Tokenliste zur Verwaltung und Erkennung des Quelltextes Erweiterung des terminalen Alphabets der zu compilierenden formalen Sprache Implementierung als endlicher Automat • Parser: kontextfreie Grammatik als Grundlage Erweiterung des Sprachumfangs Implementierung als kontextfreie Grammatik	• modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I).	Arbeitsblätter und Skript zum Compilerbau in der moodle-Plattform

Unterrichtsvorhaben Q2-III

Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

Leitfragen: Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinennahen Programms mit diesen Komponenten vorstellen? Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen?

Zeitbedarf: 10 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht.

Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Mit Hilfe einer entsprechenden Java-Methode wird plausibel, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

Lernmittel / Materialien:

- Arbeitsblätter in der moodle-Plattform
- geeigneter Modellrechner (z. B. Johnny, WinAli, Microprocessor Simulator, o. a.)

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme: • prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher • einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann • Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	geeigneter Modellrechner • Johnny • WinAli • Microprocessor Simulator
2. Grenzen der Automatisierbarkeit: • Vorstellung des Halteproblems • Unlösbarkeit des Halteproblems • Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		Arbeitsblätter von der Schulhomepage zum Halteproblem Mögliche Verknüpfung zu Unterrichtsvorhaben Q1-1a (Backracking)

Unterrichtsvorhaben Q2-IV

Thema: Modellierung und Implementierung dynamischer nichtlinearer Datenstrukturen am Beispiel der Graphen

Leitfragen: Bei welchen Problemstellungen reichen die bekannten Datenstrukturen nicht aus? Welche Möglichkeiten gibt es, flexibel miteinander verknüpfte Informationen zu verwalten? Wie hängen die Datenstrukturen Graph, Baum und Liste zusammen?

Zeitbedarf: 15 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext (z. B. das Eulerkreisproblem), in dem Daten in Form eines Graphen verwaltet werden, werden der Aufbau und die Darstellungsformen von Graphen am Beispiel dargestellt und ausgewählte Problemstellungen exemplarisch analysiert.

Die Operationen der Klasse Graph werden erläutert und im Anwendungszusammenhang bei der Lösung grundlegender Probleme (wie z.B. der Traversierung eines Graphen) genutzt.

Lernmittel / Materialien:

- Arbeitsblätter in der moodle-Plattform
- eine didaktische Entwicklungsumgebung (z. B. Java-Editor o. a.)

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur des Graphen im Anwendungskontext • Grundbegriffe (Knoten, Kanten und anderes) • Darstellungsformen eines Graphen	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	Eulerkreisproblem: • Arbeitsblätter zum Eulerweg / Eulerkreis • Visualisierungsprogramm für Graphen

	• analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • erläutern Operationen dynamischer (linearer oder/und nicht-linearer) Datenstrukturen (A),	
2. Die Datenstruktur des Graphen im Anwendungskontext unter Nutzung der Klasse Graph • Modellierung von Anwendungssituationen als Graph • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen • Erarbeitung der Funktionalität der Klasse Graph • Modellierung und Implementierung verschiedener Problemstellungen unter Verwendung der Klasse Graph.	•	Projektarbeit in einem größeren Kontext: • Navigationssystem • Weiterführung des Eulerkreisproblems Arbeitsblätter in der moodle-Plattform zu: • Tiefensuche / Breitensuche • kürzeste Wege mit Dijkstra mögliche Vertiefungen: • Traveling Salesman Problem • minimale Spannbäume

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik die folgenden fachmethodischen und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 14 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 15 bis 21 sind fachspezifisch angelegt.

<u>Überfachliche Grundsätze:</u>

1. Geeignete Problemstellungen zeichnen die Ziele des Unterrichts vor und bestimmen die Struktur der Lernprozesse.
2. Inhalt und Anforderungsniveau des Unterrichts entsprechen dem Leistungsvermögen der Schülerinnen und Schüler
3. Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4. Medien und Arbeitsmittel sind schülernah gewählt.
5. Die Schülerinnen und Schüler erreichen einen Lernzuwachs.
6. Der Unterricht fördert eine aktive Teilnahme der Schülerinnen und Schüler
7. Der Unterricht fördert die Zusammenarbeit zwischen den Schülerinnen und Schülern und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8. Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schülerinnen und Schüler
9. Die Schülerinnen und Schüler erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10. Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
11. Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12. Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13. Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14. Es herrscht ein positives pädagogisches Klima im Unterricht.

<u>Fachliche Grundsätze</u>

15. Der Unterricht unterliegt der Wissenschaftsorientierung und ist dementsprechend eng verzahnt mit seiner Bezugswissenschaft.
16. Der Unterricht ist problemorientiert und soll von realen Problemen ausgehen und sich auf solche rückbeziehen.
17. Der Unterricht folgt dem Prinzip der Exemplarizität und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten erkennen.
18. Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert und gewinnt dadurch für die Schülerinnen und Schüler an Bedeutsamkeit.
19. Der Unterricht ist handlungsorientiert, d. h. projekt- und produktorientiert angelegt.
20. Im Unterricht werden sowohl für die Schule didaktisch reduzierte als auch reale Informatiksysteme aus der Wissenschafts-, Berufs- und Lebenswelt eingesetzt.
21. Der Unterricht beinhaltet reale Begegnung mit Informatiksystemen.

2.3 Grundsätze der Leistungsbewertung

2.3.1 Transparenz der Leistungsbeurteilung

Schulische Leistungsbewertung steht im Spannungsfeld pädagogischer und gesellschaftlicher Zielsetzung.

Unter pädagogischen Gesichtspunkten hat sie vornehmlich das Individuum im Blick. Hier soll sie über den Leistungszuwachs rückmelden und dadurch die Motivation für weitere Anstrengungen erhöhen. Sie ermöglicht den Schülerinnen und Schülern ihre noch vorhandenen fachlichen Defizite wie auch ihre Stärken und Fähigkeiten zu erkennen um dadurch ein realistisches Selbstbild aufzubauen. Sie ist Basis für gezielte individuelle Förderung.

Für die Erziehungsberechtigten sind Noten eine einfache und zentrale Information zum Leistungstand ihre Kinder. Sie bieten den Anlass, über die Ursache von Defiziten und über die Beseitigung von Lernschwierigkeiten verschiedenster Art Rücksprache zu halten. Noten sind zudem Grundlage und Anlass, in den halbjährlich stattfindenden pädagogischen Konferenzen über die Schwierigkeiten und besonderen Probleme einzelner Schüler wie auch Klassen zu beraten und Maßnahmen zur Verbesserung zu beschließen.

Schulische Leistungsbewertung ist eingebettet in die durch das Schulgesetz § 48 (Grundsätze der Leistungsbewertung), APO - GOSt §13 bis §17 sowie Kapitel 3 des Kernlehrplans Informatik für die gymnasiale Oberstufe vorgegebene Grundsätze und Verfahren. Daraus erwächst für die Schulen konkret die Aufgabe, sowohl die individuellen Schwächen und Stärken der Schüler zu diagnostizieren und gegebenenfalls die Defizite durch gezielte Maßnahmen zu beseitigen sowie besondere Begabungen zu fördern.

Die gesellschaftliche Funktion von Noten zu erfüllen ist der Schule aufgegeben. Noten entscheiden mit über Schullaufbahnen, Versetzungen und Abschlüsse. Zeugnisse sind mit entscheidender Parameter bei der Zuteilung von Berufs- und Lebenschancen. Daraus erwachsen für die Beurteilenden eine besondere Verantwortung und die Pflicht einer größtmöglichen Objektivität bei der Notenfindung.

Die Fachkonferenz Informatik legt die Kriterien für die Leistungsbeurteilung fest. Die Lehrerinnen und Lehrer machen diese Kriterien den Schülerinnen und Schülern transparent.

2.3.2 Grundsätze der Leistungsbeurteilung

Es gelten folgende Grundsätze der Leistungsbewertung:

- Lernerfolgsüberprüfungen sind ein kontinuierlicher Prozess. Bewertet werden alle im Zusammenhang mit dem Unterricht erbrachten Leistungen (schriftliche Arbeiten, mündliche Beiträge, praktische Leistungen).
- Leistungsbewertung bezieht sich auf die im Unterricht geförderten Kompetenzen.
- Die Lehrperson gibt den Schülerinnen und Schülern im Unterricht hinreichend Gelegenheit, die entsprechenden Anforderungen der Leistungsbewertung im Unterricht in Umfang und Anspruch kennenzulernen und sich auf sie vorzubereiten.
- Bewertet werden der Umfang, die selbstständige und richtige Anwendung der Kenntnisse, Fähigkeiten und Fertigkeiten sowie die Art der Darstellung.

2.3.3 Formen der Leistungsüberprüfung

2.3.3.1 Kursarbeiten bzw. Klausuren

Kursarbeiten bzw. Klausuren dienen der schriftlichen Überprüfung der Lernergebnisse einer vorausgegangenen Unterrichtsreihe. Sie sind so anzulegen, dass Sachkenntnisse und methodische Fertigkeiten nachgewiesen werden können. Sie bedürfen einer angemessenen Vorbereitung und verlangen klare Aufgabenstellungen. Im Umfang und Anforderungsniveau sind Kursarbeiten bzw. Klausuren abhängig von den kontinuierlich ansteigenden Anforderungen entsprechend dem Lehrplan.

Es ist darauf zu achten, dass nicht nur die Richtigkeit der Ergebnisse und die inhaltliche Qualität, sondern auch die angemessene Form der Darstellung unabdingbare Kriterien der Bewertung der geforderten Leistung sind.

Am Ernst-Moritz-Arndt-Gymnasium Bonn werden die Kursarbeiten bzw. Klausuren in der Regel nach einem vorab festgelegten Punkteschema bewertet. Das Bewertungsraster orientiert sich dabei zunehmend an den Vorgaben für das Zentralabitur.

In der Sekundarstufe II wird spätestens in der Abiturvorklausur die im Zentralabitur gemäß unten aufgeführter Tabelle vorgegebene Zuordnung der erreichten Punkte (maximale Punktzahl: 100 im GK, 150 im LK) zur Note als Grundlage der Notenfindung genutzt.

Grundkurs (100 Pkt.)			Leistungskurs (150 Pkt.)	
Punkte	Note		Punkte	Note
0-19	6		0-29	6
20-26	5-		30-39	5-
27-32	5		40-48	5
33-38	5+		49-57	5+
39-44	4-		58-67	4-
45-49	4		68-74	4
50-54	4+		75-82	4+
55-59	3-		83-89	3-
60-64	3		90-97	3
65-69	3+		98-104	3+
70-74	2-		105-112	2-
75-79	2		113-119	2
80-84	2+		120-127	2+
85-89	1-		128-134	1-
90-94	1		135-142	1
95-100	1+		143-150	1+

Die Fachkonferenz legt die Dauer der Kursarbeiten und Klausuren fest. Am EMA Bonn gelten für die Sekundarstufe I und II folgende Regelungen:

Klasse	1. Klausur, 1. HJ	2. Klausur 1. HJ	1. Klausur 2. HJ	2. Klausur, 2. HJ
EF	---	90 min (2 h)		
GK Q1	90 min (2 h)			
GK Q2	135 min (3 h)		225 min (5 h)	---
LK Q1	135 min (3 h)			
LK Q2	225 min (5 h)		270 min (6 h)	---

In der Qualifikationsphase I kann die erste Klausur im 2. Halbjahr durch eine Facharbeit ersetzt werden.

2.3.3.2 Mitarbeit im Unterricht

Der Beurteilungsbereich „Mitarbeit im Unterricht“ erfasst die Qualität und Kontinuität der Beiträge, die die Schülerinnen und Schüler im Unterricht erbringen. Diese Beiträge sollen unterschiedliche mündliche und schriftliche Formen in enger Bindung an die Aufgabenstellung, die inhaltliche Reichweite und das Anspruchsniveau der jeweiligen Unterrichtseinheit umfassen.

Bei den mündlichen Leistungen im Unterricht sind zu bewerten:

- Beteiligung am Unterrichtsgespräch
- Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
- Präsentation von Arbeitsergebnissen
- Mitarbeit in Partner- und Gruppenarbeitsphase
- Führen eines Lerntagebuchs

Neben der Richtigkeit, Vollständigkeit und Komplexität der Gedankengänge sind die der Altersstufe angemessene sprachliche Darstellung und die Verwendung der Fachsprache von Bedeutung.

Bei der Unterrichtsgestaltung sind den Schülerinnen und Schülern hinreichend Möglichkeiten zur Mitarbeit zu eröffnen, z.B. durch

- praktische Leistungen am Computer als Werkzeug im Unterricht,
- Protokolle und Referate,
- Führen eines Lerntagebuchs,
- Projektarbeit (oft in Form von Gruppenarbeit),
- Lernerfolgsüberprüfungen und schriftliche Übungen.

2.3.3.3 Individuelle Förderung

Die Lehrerinnen und Lehrer beobachten die individuellen Leistungen in allen Bereichen der Informatik über einen längeren Zeitraum, um auf dieser Grundlage ein Leistungsbild zu erhalten. Neben der Orientierung an den Kompetenzstandards der jeweiligen Jahrgangsstufe

kann bei der Leistungsbewertung auch die jeweilige Entwicklung des Schülers bzw. der Schülerin, gemäß der zu beobachtenden Lern- und Denkfortschritte, berücksichtigt werden.

Der Informatikunterricht lebt von der verantwortungsvollen und selbständigen Arbeit der Schülerinnen und Schüler, so dass die Lehrperson die nötige Zeit hat, bei Bedarf gezielt und individuell zu fördern.

Leistungsstärkere Schülerinnen und Schüler können ihr Wissen anhand von vertiefenden Problemstellungen erweitern.

2.3.3.4 Bildung der Zeugnisnote

In die Note gehen alle im Unterricht erbrachten Leistungen ein. Dabei nehmen die Beurteilung der Kursarbeiten bzw. Klausuren den gleichen Stellenwert wie die Leistungen im Bereich der Mitarbeit im Unterricht ein. Zudem ist bei der Notenfindung die individuelle Lernentwicklung der Schülerinnen und Schüler angemessen zu berücksichtigen.

2.4 Lehr- und Lernmittel

Eingesetzte Lehrbücher und Arbeitsmaterialien:

- Skripte und Arbeitsblätter (insbesondere Leitprogramm in EF)
- Informatik, Lehrwerk für die gymnasiale Oberstufe – Neubearbeitung, Schöningh-Verlag
- Arbeitsblätter und Programmvorlagen und auf der schuleigenen Lernplattform Moodle.

Eingesetzte Software (jeweils in der aktuellen Version):

- Java SDK
- BlueJ
- Greenfoot
- Java-Editor
- WinAli
- JFlap
- xampp
- Cryptool
- GLOOP
- Weitere Demonstrationsprogramme

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Nutzung außerschulischer Lernorte

- Besuch der SchülerKrypto der Universität Bonn in der EF
- Besuch des Schülerlabors Informatik an der RWTH Aachen
- Exkursion zu verschiedenen IT-Firmen (Deutsche Post, Orbit GmbH, WetterOnline, …) im Stadtgebiet Bonn

4 Qualitätssicherung und Evaluation

Das schulinterne Curriculum stellt keine starre Größe dar, sondern ist als „lebendes Dokument“ zu betrachten. Dementsprechend sind die Inhalte stetig zu überprüfen, um ggf. Modifikationen vornehmen zu können. Die Fachkonferenz (als professionelle Lerngemeinschaft) trägt durch diesen Prozess zur Qualitätsentwicklung und damit zur Qualitätssicherung des Faches bei.

Durch Diskussion der Aufgabenstellung von Klausuren in Fachdienstbesprechungen und eine regelmäßige Erörterung der Ergebnisse von Leistungsüberprüfungen wird ein hohes Maß an fachlicher Qualitätssicherung erreicht.

Jeweils vor Beginn eines neuen Schuljahres werden in einer Sitzung der Fachkonferenz für die nachfolgenden Jahrgänge zwingend erforderlich erscheinende Veränderungen diskutiert und ggf. beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.