GYMNASIUM SIEGBURG ALLEESTRASSE

ABIBAC-SCHULE • CERTILINGUA • MUSIKPROFIL

Gymnasium Siegburg Alleestraße

Schulinterner Lehrplan

Informatik

Stand: 06.05.2015

1 Die Fachgruppe Informatik am Gymnasium Siegburg Alleestraße

Das Fach Informatik wird am Gymnasium Siegburg Alleestraße ab der Jahrgangsstufe 8 im Rahmen des Wahlpflichtbereichs unterrichtet. In diesem zweijährigen Kurs werden in altersstufengerechter Weise beispielsweise Grundlagen der Algorithmik am Beispiel verschiedener didaktischer Lernumgebungen (Scratch, Niki) sowie die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen thematisiert.

Im Rahmen der Informationstechnischen Grundbildung erhalten die Schülerinnen und Schüler der Jahrgangsstufe 6 einen Einblick in die grundlegende Funktionsweise von Rechner- und Dateisystemen, erlernen den grundlegenden Umgang mit Textverarbeitungs- und Präsentationssoftware, erkennen und diskutieren Gefahren im Internet.

In der Sekundarstufe II wird Informatik ebenfalls als Wahlfach (Grundkurs) angeboten. Um insbesondere Schülerinnen und Schülern, die in der Sekundarstufe I keinen Informatikunterricht besucht haben, gerecht zu werden, wird in den Kursen der Oberstufe besonderer Wert darauf gelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Bestehen der Kurse 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.

Zurzeit besteht die Fachschaft Informatik des GSA aus vier Lehrkräften, von denen drei die Sek-II-Lehrbefähigung aufweisen.

- Fachvorsitzende: Antje Waßmann
- Stellvertreter: Dr. Verena Gondek

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: Im "Übersichtsraster Unterrichtsvorhaben" (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Es verschafft 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 grober Orientierungswert, der bei

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. 80 Prozent der Unterrichtszeit 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. Abweichungen von den vorgeschlagenen Vorgehensweisen bezüglich der konkretisierten Unterrichtsvorhaben sind im Rahmen der pädagogischen Freiheit der Lehrkräfte jederzeit möglich. 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

Im Folgenden sollen Unterrichtsvorhaben für das Fach Informatik dargestellt werden. 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.

I) Einführungsphase – Grundkurs

Unterrichtsvorhaben Nr. 1	**Unterrichtsvorhaben Nr. 2**
Thema: Überblick über das Fach Informatik, Grundlagen von Informatiksystemen und Geschichte der Informatik Zentrale Kompetenzen: Argumentieren (A), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Informatiksysteme Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Digitalisierung, Einzelrechner, Dateisysteme Internet, Einsatz von Informatiksystemen	Thema: Grundlagen der Programmierung mit Java und einfache Algorithmik Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen, Formale Sprachen und Automaten, Informatiksysteme Inhaltliche Schwerpunkte: Analyse, Entwurf und Implementierung einfacher

Wirkungen der Automatisierung Geschichte der automatischen Datenverarbeitung Zeitbedarf: 6 – 9 Stunden	Algorithmen Syntax und Semantik einer Programmiersprache Zeitbedarf: 30 – 33 Stunden
Unterrichtsvorhaben Nr. 3 Thema: Objektorientierte Modellierung und Implementierung von Klassen- und Objektbeziehungen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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: 21 – 24 Stunden	**Unterrichtsvorhaben Nr. 4** Thema: Arrays und Entwurf sowie Implementierung einfacher Such- und Sortierverfahren Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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: 10 – 12 Stunden
Unterrichtsvorhaben Nr. 5 Thema: Vertiefung von Objektbeziehungen anhand von grafischen Spielen, Simulationen und grafischen Oberflächen Zentrale Kompetenzen: Modellieren (M), Implementieren (I), Kommunizieren und Kooperieren (K) 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: 22 Stunden	

I) Qualifikationsphase – Grundkurs/Leistungskurs

Die Aufstellung der Unterrichtsvorhaben erfolgt für den Grundkurs und den Leistungskurs kombiniert, da diese bei Bedarf als Huckepack-Kurs durchgeführt werden.

Qualifikationsphase I

Grundkurs	Zusätzlich im Leistungskurs
Unterrichtsvorhaben QI-1a Thema: Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen, Formale Sprachen und Automaten, Informatiksysteme Inhaltliche Schwerpunkte: Objekte und Klassen, Analyse, Entwurf und Implementierung von Algorithmen, Syntax und Semantik einer Programmiersprache Zeitbedarf: 6 – 9 Stunden	**Unterrichtsvorhaben QI-1b** Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit Zentrale Kompetenzen: Argumentieren (A), Kommunizieren und Kooperieren (K) Inhaltsfelder: Informatiksysteme, Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Einzelrechner und Rechnernetze Grenzen der Automatisierung Zeitbedarf: 9 -12 Stunden
Unterrichtsvorhaben QI-2a Thema: Rekursive Algorithmen und Backtracking in Anwendungskontexten Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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 Zeitbedarf: 18 (27) Stunden	
Unterrichtsvorhaben QI-3a Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen, Formale Sprachen und Automaten Inhaltliche Schwerpunkte: Objekte und Klassen Analyse, Entwurf und Implementierung von Algorithmen Algorithmen in informatischen Kontexten Syntax und Semantik einer Programmiersprache Zeitbedarf**:** 20 – 23 Stunden	
Unterrichtsvorhaben QI-4a Thema: Suchen und Sortieren auf linearen Datenstrukturen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen, Formale Sprachen und Automaten Inhaltliche Schwerpunkte: Analyse, Entwurf und Implementierung einfacher Algorithmen in ausgewählten Kontexten Syntax und Semantik einer Programmiersprache Zeitbedarf: 15 - 18 (21 – 24) Stunden	

Unterrichtsvorhaben QI-5a	**Unterrichtsvorhaben QI-5b**
Thema: Endliche Automaten und formale Sprachen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Formale Sprachen und Automaten Inhaltliche Schwerpunkte: Endliche Automaten (und Kellerautomaten) Grammatiken regulärer (und kontextfreier) Sprachen Möglichkeiten und Grenzen formaler Sprachen und Automaten Algorithmen zum Suchen und Sortieren Zeitbedarf: 18 - 21 (24 – 27) Stunden	Thema: Schritte eines Compilers einer formalen Sprache Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen Formale Sprachen und Automaten, Informatiksysteme Inhaltliche Schwerpunkte: Endliche Automaten Grammatiken regulärer Sprachen Scanner, Parser und Interpreter für eine reguläre Sprache Nutzung von Informatiksystemen Zeitbedarf: 12 Stunden

Qualifikationsphase II

Grundkurs	**Zusätzlich im Leistungskurs**
Unterrichtsvorhaben QII-1a Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Daten und ihre Strukturierung, Algorithmen, Formale Sprachen und Automaten, Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Datenbanken, Algorithmen in ausgewählten informatischen Kontexten, Syntax und Semantik einer Programmiersprache, Sicherheit Zeitbedarf: 18-21 (24 – 27) Stunden	

Unterrichtsvorhaben QII-2a Thema: Sicherheit und Datenschutz in Netzstrukturen Zentrale Kompetenzen: Argumentieren (A), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) Inhaltsfelder: Informatiksysteme, Informatik, Mensch und Gesellschaft Inhaltliche Schwerpunkte: Einzelrechner und Rechnernetze Sicherheit Nutzung von Informatiksystemen, Wirkungen der Automatisierung Zeitbedarf: 9 - 12 Stunden	**Unterrichtsvorhaben QII-2b** Thema: Modellierung und Implementierung von Client-Server-Anwendungen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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 Einzelrechner und Rechnernetzwerke Nutzung von Informatiksystemen Zeitbedarf: 10 Stunden
Unterrichtsvorhaben QII-3a Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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 Zeitbedarf: 15 -18 (19 - 23) Stunden	**Unterrichtsvorhaben QII-3b** Thema: Modellierung und Implementierung dynamischer nichtlinearer Datenstrukturen am Beispiel der Graphen Zentrale Kompetenzen: Argumentieren (A), Modellieren (M), Implementieren (I), Darstellen und Interpretieren (D), Kommunizieren und Kooperieren (K) 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

2.1.2 Konkretisierte Unterrichtsvorhaben

Hinweis: Thema, Inhaltsfelder, inhaltliche Schwerpunkte, Kompetenzen und Absprachen zur Konkretisierung hat die Fachkonferenz verbindlich vereinbart. In allen anderen Bereichen (Unterrichtssequenzen, verwendete 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. Solche Absprachen können darüber hinaus und bei Bedarf abweichend auch für konkrete Unterrichtsvorhaben vorgenommen werden.

I) Einführungsphase – Grundkurs

Im Folgenden werden die im Übersichtsraster dargestellten Unterrichtsvorhaben konkretisiert. Dies hat lediglich 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. D.h. die 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: 6 - 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 SuS 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 Fronter
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 Datenverarbeitung (A).	z.B. ASCII-Code

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: 30 - 33 Stunden

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

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Einführung in die Programmierung - Grundlagen - Programmaufbau - Datentypen und Operationen - Variablen - Bedingte Verzweigungen - Schleifen - Methoden	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • 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).	Entweder Leitprogramm „Einführung in die Programmierung“ (Werkzeuge: JavaEditor) oder Einstieg in die Programmierung mit Hilfe einer grafischen Umgebung, z.B. Spinnwelt oder Roboter (Werkzeug: Greenfoot) Struktogramme Lehrbuch: Informatik I, Schöningh

Unterrichtsvorhaben Nr. 3

Thema: Objektorientierte Modellierung und Implementierung von Klassen- und Objektbeziehungen

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

Zeitbedarf: 21 - 24 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Ausgehend von der Entwicklungsumgebung BlueJ wird den Schülerinnen und Schüler die Bedeutung von Objekten und Klassen deutlich gemacht. Sie modellieren eigene Klassen und differenzieren diese in ihren Attributen und Methoden. Neben der Erzeugung von Objekten erlernen sie auch deren Benutzung und Interaktionen untereinander.

Klassen werden schrittweise durch Vererbung konkretisiert und deren Funktionen implementiert. Dabei werden die bereits erlernten Grunddatentypen und verschiedenen Arten von Methoden in anderem Zusammenhang angewendet und eingeübt. Insbesondere wird das Geheimnisprinzip vertieft.

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Grundlagen der Entwicklungsumgebung BlueJ und Einführung in die OOP a) Was ist ein Objekt? Was ist eine Klasse? b) Objekte erzeugen c) Darstellung / Verhalten von Objekten d) Methoden und Attribute e) Objektinteraktion f) Geheimnisprinzip g) Vererbung	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 Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von 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).	BlueJ Fronter Lehrbuch: Informatik I, Schöningh Leitprogramm „Vererbung und Polymorphie“ UML-Diagramme (z.B. mit UMLed)

Unterrichtsvorhaben Nr. 4

Thema: Arrays und Entwurf sowie Implementierung einfacher Such- und Sortierverfahren

Leitfragen: Wie kann man Arrays zum Sortieren von großen Datenmengen benutzen? Wie effizient ist ein Algorithmus hinsichtlich seines Laufzeitverhaltens?

Zeitbedarf: 10 - 12 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:
Es werden zunächst die Grundlagen im Umgang mit Arrays erarbeitet. Aufbauend darauf können die SuS ihr Wissen kontextbezogen anwenden:

Dazu werden sowohl lineare und binäre Suche als auch mindestens zwei der einfachen Sortierverfahren behandelt, indem sie simuliert, implementiert und getestet werden. In der anschließenden Analyse reflektieren die SuS das Laufzeitverhalten der Algorithmen im best, worst und average case.

Optional kann zusätzlich ein einfaches kryptologisches Verfahren (z.B. CAESAR) behandelt werden. Neben der Durchführung der einfachen Verschlüsselungstechnik werden dabei 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).	Fronter MathePrisma Uni Wuppertal Youtube BlueJ
2. Optional: 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).	Fronter MathePrisma Uni Wuppertal BlueJ

Unterrichtsvorhaben Nr. 5

Thema: Vertiefung von Objektbeziehungen anhand von grafischen Spielen, Simulationen und grafischen Oberflächen

Leitfragen: Wie kann ein umfangreicheres Programmierprojekt angemessen realisiert werden (von der Modellierung zur Implementierung)? Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche realisieren?

Zeitbedarf: 22 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
Projekt: Erstellung einer eigenen Animation/eines eigenen Spiels a) Entwicklung einer Spiel-/Animationsidee (Anforderungsdefinition) b) Modellierung der benötigten Strukturen c) Implementierung der Idee unter Verwendung von Hilfestellungen aus dem Internet d) Dokumentation des Projekts	Die Schülerinnen und Schüler • konstruieren zu kontextbezogenen Problemstellungen informatische Modelle (M). • implementieren auf der Grundlage von Modellen oder Modellausschnitten Computerprogramme (I) • testen und korrigieren Computerprogramme (I). • kommunizieren und kooperieren in Gruppen- und Partnerarbeit (K). • präsentieren Arbeitsabläufe und Ergebnisse(K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	BlueJ mit Gloop oder Greenfoot Fronter Lerntagebuch

II) Lehrplan für die Qualifikationsphase – Grundkurs/Leistungskurs

Die Aufstellung der Unterrichtsvorhaben erfolgt für den Grundkurs und den Leistungskurs kombiniert, da diese bei Bedarf als Huckepack-Kurs durchgeführt werden.

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
- 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).

Unterrichtsvorhaben QI-1a

Thema: Wiederholung der objektorientierten Modellierung und Programmierung

Leitfragen: Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?

Zeitbedarf: 6 - 9 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt sein, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann. Um die Aufgabe einzugrenzen, können (nach der ersten Problemanalyse) einige Teile (Modellierungen oder Teile von Java-Klassen) vorgegeben werden.

Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Exemplarisch wird eine Klasse dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die

Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels 1. Analyse der Problemstellung 2. Analyse der Modellierung (Implementationsdiagramm) 3. Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) 4. Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) 5. Dokumentation von Klassen 6. Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D).	*Beispiel*: Tannenbaum Ein Tannenbaum soll mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren. Ein Prototyp, der bereits mit Kugeln geschmückt werden kann, kann zur Verfügung gestellt werden. Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass entsprechende Methoden in einer gemeinsamen Oberklasse realisiert werden.

Unterrichtsvorhaben QI-1b (nur Leistungskurs, Ausnahme: von-Neumann-Architektur)

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: 9 - 12 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht.

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.

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien oder Materialien**
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme (auch im GK) a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c) 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).	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem

Unterrichtsvorhaben QI-2a

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: 18 (27) 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.

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)
3. NP-vollständige Probleme lösen mit Backtracking (nur Leistungskurs) • Erarbeitung verschiedener NP-vollständiger Probleme • Algorithmische Beschreibung	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I),	• Rucksackproblem • Irrgartenproblem • Projektarbeit zu einer Problemstellung

einer Lösungsidee • Implementierung eines Problems als Backtrackingalgorithmus	• 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).	

Unterrichtsvorhaben QI-3a

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

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

Zeitbedarf: 20-23 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse `Queue` 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 Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse `List` eingeführt und in einem Anwendungskontext verwendet. In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt. Im Leistungskurs liegt ein zusätzlicher Schwerpunkt auf der eigenständigen Implementierung verschiedener Methoden.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),	*Beispiel:* Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger)

• Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen • Erarbeitung der Funktionalität der Klasse Queue • Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue • Implementierung der Klasse Queue (nur LK) **2. 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 eines oder mehrerer Objekte der Klasse Stack • Implementierung der Klasse Stack (nur LK) **3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List** • Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits • Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List. **4. Vertiefung - Anwendungen von Listen, Stapeln oder**	• analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D). • implementieren Operationen dynamischer linearer Datenstrukturen (I)	Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ eines Patienten und das „Entfernen“ eines Patienten, wenn er zur Behandlung gerufen wird. Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet. *Beispiel*: Terme in Postfix-Notation Die sog. UPN (*Umgekehrt-Polnische-Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet werden.

Schlangen in mindestens einem weiteren Kontext		z.B. Priority Queue

Unterrichtsvorhaben QI-4a

Thema: Suchen und Sortieren auf linearen Datenstrukturen

Leitfragen: *Wie kann man gespeicherte Informationen günstig (wieder-)finden?*

Zeitbedarf: 15 - 18 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei neben einem iterativen auch ein rekursives Verfahren thematisiert wird und mindestens ein Verfahren selbst entwickelt und implementiert wird. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarf und Zahl der Vergleichsoperationen miteinander verglichen.

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.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Suchen von Daten in Listen und Arrays • Lineare Suche in Listen und in Arrays • Binäre Suche in Arrays als Beispiel für rekursives Problemlösen • Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf) **2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren**	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M) und "Backtracking" • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive	*Beispiel:* Karteiverwaltung Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. oder *Beispiel*: Bundesjugendspiele Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das

• Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste • Implementierung eines einfachen Sortierverfahrens für ein Feld • Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen) **3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen** • Grafische Veranschaulichung der Sortierverfahren • Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarfs bei beiden Sortierverfahren • Beurteilung der Effizienz der beiden Sortierverfahren	Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • 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) • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	Modell auf die drei Disziplinen „Lauf”, „Sprung“ und „Wurf“ beschränken. Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können.

Unterrichtsvorhaben QI-5a

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, endlichen Automaten und regulären Grammatiken?*

Zeitbedarf: 18 – 21 (24 - 27) Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel wird ein nichtdeterministischer Akzeptor eingeführt als Alternative gegenüber einem entsprechenden deterministischen Akzeptor.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert.

Auch andere Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

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.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Endliche Automaten • Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten • Untersuchung, Darstellung und Entwicklung endlicher Automaten **2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen** • Erarbeitung der formalen Darstellung regulärer Grammatiken • Untersuchung, Modifikation und Entwicklung von Grammatiken • Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden • Entwicklung regulärer Grammatiken zu endlichen Automaten **3. Grenzen endlicher Automaten**	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten (M), • modifizieren Grammatiken regulärer oder kontextfreier Sprachen (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D). • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).	*Beispiel 1:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Beispiel 2:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Beispiel 3:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$

4. Zusammenhang zwischen Kellerautomaten und kontextfreien Grammatiken (nur Leisungskurs) • kontextfreie Grammatik: Definition, Anwendungen • Modell des Kellerautomaten Definition Darstellungsformen Anwendungen / Sprache eines Kellerautomaten • Zusammenhang zwischen Kellerautomaten/kontextfreien Grammatiken • Grenzen der Kellerautomaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften von Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein Kellerautomat akzeptiert (D), • entwickeln und modifizieren zu einer Problemstellung Kellerautomaten (M), • entwickeln zur Grammatik einer kontextfreien Sprache einen zugehörigen Kellerautomaten (M), • analysieren und erläutern Grammatiken kontextfreier Sprachen (A), • modifizieren Grammatiken kontextfreier Sprachen (M), • entwickeln zu einer kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M),	

Unterrichtsvorhaben QI-5b (nur Leistungskurs)

Thema: Schritte eines Compilers einer formalen Sprache

Leitfragen: Wie funktioniert ein Compiler genau? Wie erkennt ein Compiler die Bestandteile einer Programmiersprache? Wie erkennt er die syntaktische Korrektheit?

Zeitbedarf: 12 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

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).

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Die Schritte eines Compilers <u>Scanner:</u> • 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 <u>Parser:</u> • kontextfreie Grammatik als Grundlage • Erweiterung des Sprachumfangs • Implementierung als kontextfreie Grammatik	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), • 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), • 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). • 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), • 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), • modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I).	

Unterrichtsvorhaben QII-1a

Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

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

Zeitbedarf: 18 – 21 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Ausgehend von einer vorhandenen Datenbank 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.

In anderen Anwendungskontexten müssen Datenbanken erst noch 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 Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitäten inklusive ihrer Attribute und Relationen in einem vorgegebenen Datenbankschema darzustellen.

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.

Unterrichtssequenzen	zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Nutzung von relationalen Datenbanken Aufbau von Datenbanken und Grundbegriffe: • Entwicklung von Fragestellungen zur vorhandenen Datenbank und • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe: Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema 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 **2. 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	Die Schülerinnen und Schüler • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D). • implementieren ein relationales Datenbankschema als Datenbank (I),	*Beispiel:* VideoCenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontends zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunter zu laden und lokal zu installieren. Unter http://dokumentation.videocenter.schule.de/old/video/index.html – SQL-Zoo – Datenbankserver, z.B. XAMPP

• Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln <u>Redundanz, Konsistenz und Normalformen</u> • 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)		

Unterrichtsvorhaben QII-2a

<u>Thema:</u> Sicherheit und Datenschutz in Netzstrukturen

<u>Leitfragen:</u> *Wie werden Daten in Netzwerken übermittelt? Was sollte man in Bezug auf die Sicherheit beachten?*

<u>Zeitbedarf:</u> 9 - 12 Stunden

<u>Absprachen zur vorhabenbezogene Konkretisierung:</u>

Anschließend an das vorhergehende Unterrichtsvorhaben zum Thema Datenbanken werden der Datenbankzugriff aus dem Netz, Topologien von Netzwerken, eine Client-Server-Struktur, das TCP/IP-Schichtenmodell sowie Sicherheitsaspekte beim Zugriff auf Datenbanken und verschiedene symmetrische und asymmetrische kryptografische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken • Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A),	

Datenbankzugriffs • Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz • Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen **2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht**	• untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen 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), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	

Unterrichtsvorhaben QII-2b

Thema: Modellierung und Implementierung von Client-Server-Anwendungen

Zeitbedarf: 10 Stunden

Absprachen zur vorhabenbezogene Konkretisierung:

Ausgehend von einer einfachen Echo-Anwendung werden 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.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Entwicklung einer Echo-Anwendung • Grundlegende Begriffe • ZA-Klassen	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und ihren	
2. Entwicklung einer Chat-Anwendung		

• Nutzung der Datenstruktur List	Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • 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), • 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). • modifizieren Algorithmen und Programme (I), • 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 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). • 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), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M).	
3. Entwicklung einer beliebigen Client-Server-Anwendung		Umsetzung eines größeren Projektes • Netzwerkspiel (wie z.B. Tic Tac Toe) • Anwendungsserver (wie z.B. Buchungsserver für Kinokarten)

Unterrichtsvorhaben QII-3a

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

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: 15 - 18 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 Materialien für das Zentralabitur in NRW) 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 Umkehrbarbeit 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 Materialien für das Zentralabitur in NRW) weitere Klassen oder Methoden in diesem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

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

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Analyse von Baumstrukturen in verschiedenen Kontexten (a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene,	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern	*Beispiel:* Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. oder

Vollständigkeit) (b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	Algorithmen und Programme (A),	*Beispiel*: Ahnenbaum Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Weitere Beispiele für Anwendungskontexte für binäre Bäume:* *Beispiel*: Suchbäume (zur sortierten Speicherung von Daten) Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) *oder* *Beispiel*: Entscheidungsbäume Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. Die Fragen, die möglich sind, wenn die Antwort auf eine Frage mit „ja“ beantwortet wird, befinden sich im linken Teilbaum, die Fragen, die möglich sind, wenn die Antwort „nein“ lautet, stehen im rechten Teilbaum. *oder* *Beispiel*: Codierungsbäume für Codierungen, deren Alphabet aus genau zwei Zeichen besteht Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Wenn man im Gesamtbaum startet

	• beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),	und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens.
2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms (c) Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen (d) Implementierung ausgewählter Standardoperationen der Klasse BinarySearchTree (e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf	• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M) und Backtracking, • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),	*Beispiel*: Informatikerbaum *als binärer Baum* In einem *binären Baum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Modellierung eines Entwurfsdiagramms und	• modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I),	*Beispiel*: Informatikerbaum als *Suchbaum* In einem binären *Suchbaum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem

Entwicklung eines Implementationsdiagramm, grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften (c) Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation (d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums	• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • implementieren Operationen dynamischer nicht-linearer Datenstrukturen (I),	Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		*Beispiel:* Codierungsbäume (s.o.) oder Huffman-Codierung

Unterrichtsvorhaben QII-3b (nur Leistungskurs)

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.

<table>
<tr><th>Unterrichtssequenzen</th><th>zu entwickelnde Kompetenzen</th><th>Beispiele, Medien, Materialien</th></tr>
<tr><td>1. Die Datenstruktur des Graphen im Anwendungskontext
• Grundbegriffe (Knoten, Kanten und anderes)
• Darstellungsformen eines Graphen</td><td rowspan="2">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).
• 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),</td><td>Eulerkreisproblem:
• Arbeitsblätter zum Eulerweg / Eulerkreis
• Visualisierungsprogramm für Graphen</td></tr>
<tr><td>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.</td><td>Projektarbeit in einem größeren Kontext:
• Navigationssystem
• Weiterführung des Eulerkreisproblems</td></tr>
</table>

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.

Überfachliche Grundsätze:

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.

Fachliche Grundsätze

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 zu 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 Leistungsstand 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.

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

Die Fachkonferenz legt die Dauer der Kursarbeiten und Klausuren fest. Am Gymnasium Siegburg Alleestraße gelten für die Sekundarstufe II folgende Regelungen:

Klasse	1. Klausur, 1. HJ	2. Klausur 1. HJ	1. Klausur 2. HJ	2. Klausur, 2. HJ
EF	---	90 min	90 min	90 min
Q1 GK	90 min	90 min	90 min	90 min
Q1 LK	135 min	135 min	135 min	135 min
Q2 GK	135 min	135 min	180 min	---
Q2 LK	180 min	255 min	255 min	---

In der Einführungsphase wird im ersten Halbjahr nur eine Klausur geschrieben. 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:

- Lehrbuch der Einführungsphase: Informatik 1 – Softwareentwicklung mit Greenfoot und BlueJ, Schöningh
- Lehrbücher der Qulifikationsphase: Informatik 2 – Modellierung, Datensturkturen und Algorithmen, Schöningh / Informatik 3 – Netzwerkanwendungen, Informatik und Gesellschaft, Datenbanken und Theoretische Informatik, Schöningh
- Skripte und Arbeitsblätter (insbesondere Leitprogramm in EF)

Eingesetzte Software (jeweils in der aktuellen Version):

- Java SDK
- BlueJ
- Greenfoot
- JFlap
- Cryptool
- GLOOP
- Weitere Demonstrationsprogramme

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Mögliche Nutzung außerschulischer Lernorte, sofern dies mit anderen schulischen Aktivitäten vereinbar ist:

- Besuch der SchülerKrypto der Universität Bonn in der EF
- Besuch des Schülerlabors Informatik an der RWTH Aachen
- Besuch des Arithmeums in Bonn
- 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.

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Jeweils vor Beginn eines neuen Schuljahres, d.h. erstmalig nach Ende der Einführungsphase im Sommer 2015 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.

Nach Abschluss des Abiturs 2017 wird eine Arbeitsgruppe aus den zu diesem Zeitpunkt in der gymnasialen Oberstufe unterrichtenden Lehrkräften auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.