Saarland
Ministerium für Bildung, Familie, Frauen und Kultur

Gymnasiale Oberstufe Saar (GOS)

Lehrplan Informatik

G-Kurs (vierstündig)

Februar 2008

LEHRPLAN FÜR DEN VIERSTÜNDIGEN G-KURS INFORMATIK IN DER HAUPTPHASE DER GYMNASIALEN OBERSTUFE

Vorbemerkung

1. und 2. Kurshalbjahr

Die zentralen Themen im Lehrplan für die Kursphase sind die Ideen des Algorithmus und der Modularisierung von Programmen und weiterführende Konzepte zu Datenstrukturen. Dazu wird an das Wissen aus der Einführungsphase angeknüpft und in einem ersten Abschnitt das strukturierte Vorgehen bei der Realisierung komplexer Algorithmen thematisiert. Die Notwendigkeit flexibler, der Problemstellung angepasster Datenstrukturen führt zur Konstruktion dynamischer Datenstrukturen und zum Konzept des abstrakten Datentyps.

Die Umsetzung der Modelle in Programme erfordert vertiefte Kenntnisse einer Programmiersprache. Im Unterricht können Delphi/Pascal oder Java benutzt werden. Bei der Gestaltung von Benutzeroberflächen sollte auch auf Aspekte der Benutzungsfreundlichkeit eingegangen werden.

Unabhängig von der Wahl der Sprache sollen die Strategien beim objektorientierten Entwurf und die Vorgehensweise bei der Implementierung objektorientierter Modelle in Programmen aufgezeigt werden.

Die Reihenfolge der Themen im Lehrplan ist ein Vorschlag, dem Lehrenden bleibt die zeitliche Anordnung der Themen freigestellt. Eine thematisch verzahnte Vorgehensweise ist didaktisch empfehlenswert.

Beim abschließenden Thema „Suchen und Sortieren“ sollen Algorithmen unterschiedlicher Komplexität zur Lösung eines Problems vorgestellt und verglichen werden. Im Kapitel „Algorithmenanalyse“ werden die hier gewonnenen Erkenntnisse aufgegriffen und dienen dort zur Begründung und Veranschaulichung von Komplexitätsbetrachtungen.

3. und 4. Kurshalbjahr

Im 2. Kurshalbjahr haben die Schülerinnen Sortieralgorithmen unterschiedlicher Effizienz kennen gelernt. Dies motiviert zu Beginn des 3. Kurshalbjahres eine tiefer gehende Beschäftigung mit der Algorithmenanalyse und der Klassifizierung von Algorithmen gemäß ihrer Effizienz. An Beispielen werden die praktischen und die prinzipiellen theoretischen Grenzen der Berechenbarkeit verdeutlicht.

Einen einführenden Zugang zu den theoretischen Grundlagen der Informatik ermöglicht die Untersuchung endlicher Automaten und einfacher Klassen formaler Sprachen. Endliche Automaten werden als abstrakte Modelle zur Beschreibung von Abläufen und der Funktionsweise von Maschinen eingeführt. Der Zusammenhang zwischen Zeichenmengen und sie erkennender Automaten führt zur Definition formaler Sprachen. Am Beispiel der regulären und der kontextfreien Sprachen werden die praktische Bedeutung formaler Sprachen bei der Kommunikation zwischen Mensch und Maschine sowie ihre Beschreibungsmöglichkeiten erklärt.

Die Darstellung des prinzipiellen Aufbaus und der Funktionsweise eines Rechners soll Verständnis für die technische Realisierung von Rechnersystemen vermitteln. Eine einfache Registermaschine dient dabei als Modell zur Beschreibung der Abläufe in einem Prozessor und zur Einführung in die Programmierung auf Maschinenebene.

Die Themen Kommunikation und Sicherheit in Rechnernetzen sollen elementares Grundwissen über die Funktionsweise von Netzwerken und damit verbundene Sicherheitsfragen vermitteln. Kryptographische Maßnahmen zum Schutz der Nachrichtenübermittlung in Netzwerken werden am Beispiel des RSA-Verfahens dargestellt.

Stoffverteilungsplan

G-Fach Informatik 1. Halbjahr der Hauptphase	(4 Wochenstunden)
Strukturiertes Programmieren	20
Objektorientiertes Modellieren und Programmieren	20

G-Fach Informatik 2. Halbjahr der Hauptphase	(4 Wochenstunden)
Datentypen und Datenstrukturen	30
Suchen und Sortieren	10

G-Fach Informatik 3. Halbjahr der Hauptphase	(4 Wochenstunden)
Algorithmenanalyse, Grenzen der Berechenbarkeit	10
Automaten und formale Sprachen	20

G-Fach Informatik 4. Halbjahr der Hauptphase	(4 Wochenstunden)
Funktionsweise von Computersystemen	15
Kommunikation und Sicherheit in Rechnernetzen	15

Informatik, G-Kurs (vierstündig)	1. Halbjahr der Hauptphase
Strukturiertes Programmieren	**20 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Prozeduren und Funktionen	
Deklaration: Formale Parameter, Parameterlisten, Signaturen Aufruf: Aktuelle Parameter, Varianten der Parameterübergabe (call by value, call by reference) Gültigkeitsbereiche von Variablen und Parametern	Algorithmen zur Lösung eines Problems werden als Prozeduren oder Funktionen (Methoden) formuliert. Nach der Implementierung genügen das Verständnis der Funktionsweise und die Kenntnis der Schnittstelle (Signatur) beim Einsatz in komplexen Problemlösungen.
Modularisierung und Strukturierung von komplexen Problemlösungen	Komplexe Probleme werden in Teilprobleme zerlegt, die klar voneinander abgegrenzt sind und getrennt bearbeitet und gelöst werden können. Die einzelnen Lösungsmodule werden zur Gesamtlösung zusammengesetzt.
Rekursive Algorithmen	
Rekursive Formulierung von Algorithmen Rekursive Unterprogramme und Funktionen: rekursiver Aufruf, Abbruchbedingung und Terminierung, Aufrufsequenz Rekursionsarten: direkt, indirekt Rekursion und Iteration, Endrekursion	Rekursion als Problemlösungsstrategie Veranschaulichung der Rekursionsidee und des Rekursionsablaufes durch Aufrufschemata. Vergleich äquivalenter rekursiver und iterativer Varianten von Algorithmen zur gleichen Problemlösung.
	Literatur/Medien Baumann R.: Informatik für die Sekundarstufe II, Klett-Verlag Stuttgart 1992 Sedgewick R.: Algorithmen, Addison-Wesley 1992 Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Aho A./Hopcroft J./Ullman J.:Data Structures and Algorithms, Addison-Wesley 1987

Informatik, G-Kurs (vierstündig)	1. Halbjahr der Hauptphase
Objektorientiertes Modellieren und Programmieren	**20 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Basiskonzepte der objektorientierten Programmierung Klassen: Datenfelder (Attribute) und Operationen (Methoden) Objekte als Klasseninstanzen Konstruktoren Methoden zur Abfrage und zur Änderung von Datenfeldern Kapselung von Daten, Geheimnisprinzip	Die Trennung von Daten und Algorithmen, die auf diesen Daten operieren, wird durch das Konzept der Objektorientierung aufgehoben. Die Eigenschaften von Gegenständen des Interesses werden durch Datenfelder, ihr Verhalten durch Methoden beschrieben. Mengen von Objekten mit gleichartigen Eigenschaften und gleichem Verhalten bilden Klassen. Die gemeinsame Eigenschaften und Verhaltensweisen werden in der Klassendeklaration beschrieben.
Objektorientiertes Modellieren und Programmieren Modellierung von Klassen und einfachen statischen Beziehungen Klassenhierarchien durch Vererbung	Beim objektorientierten Modellieren werden Klassen gleichartiger Objekte identifiziert und Beziehungen zwischen den Klassen erkannt; die Ergebnisse dieser objektorientierten Analyse werden graphisch veranschaulicht. Zur Darstellung einfacher statischer Beziehungen (hat – kennt – ist) zwischen Klassen genügen im Unterricht Klassendiagramme, wie sie in der UML definiert sind.
Realisierung von objektorientiert modellierten Problemlösungen	Ideen und Konzepte der objektorientierten Programmierung sollen durchgängig sowohl beim Modellieren und Implementieren abstrakter Datentypen, als auch bei der Bearbeitung kleiner, thematisch begrenzter Programmierprojekte demonstriert und eingeübt werden. **Literatur/Medien:** Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Eckard Modrow: Informatik mit Delphi 1/2 , Verlag emu online Bernard Schriek:Informatik mit Java /II, Nili-Verlag Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1998

Informatik, G-Kurs (vierstündig)	2. Halbjahr der Hauptphase
Datentypen und Datenstrukturen	**30 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Datentypen Einfache Datentypen	Bei der „Programmierung im Kleinen“ genügen meist die einfachen (primitiven) vordefinierten Datentypen der Programmiersprache. Bei der Modellierung komplexer Gegenstände und Zusammenhänge der Realwelt hilft die Einführung eigener, dem Problem angepasster Typen.
Datenstrukturen	
Statische Strukturen Verbunde (Records) Ein- und zweidimensionale Reihungen (Felder, Arrays)	Zusammengesetzte Datentypen ermöglichen die Zusammenfassung von Daten gleichen oder unterschiedlichen Typs.
Grundlegende Eigenschaften dynamischer Datenstrukturen Unterschied statische – dynamische Struktur; Vor- und Nachteile im Vergleich Speicherorganisation und Verwaltung Zeigerkonzept (Referenz, Referenzvariable)	Die Datenhaltung von Werten der verschiedenen Datentypen ist statisch oder dynamisch organisiert. Dynamische Datenstrukturen werden aus statischen Datenstrukturen zusammengebaut. Ihre variable Größe liefert die Flexibilität für den Aufbau beliebig komplexer Strukturen.
Spezielle dynamische Datenstrukturen	
Listen Sequentielle Anordnung von Daten Graphisches Modell, elementare Operationen (Einfügen, Lesen, Entfernen) am graphischen Modell Rekursive Definition Darstellung von Listen als Reihung oder als verkettete Liste von Knoten Listenknoten als Zusammenfassung von Datenelement (zu speichernder Wert) und Verwaltungselement (Zeiger, Referenz) Realisierung der elementaren Operationen in verketteten Listen Einfach und doppelt verkettete Listen	Listen realisieren das mathematische Modell einer endlichen Folge (Sequenz) variabler Länge von Werten gleichen Typs. Sie sind eine grundlegende Datenstruktur, mit der Spezialisierungen wie Stapel oder Schlange oder mehrdimensionale Strukturen wie Graphen und Bäume realisiert werden können. Der Verwaltung sequentieller dynamischer Strukturen im Speicher des Rechners entspricht das Modell der aus verketteten Knoten aufgebauten „verlinkten Liste“ von Daten. **Literatur/Medien** Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Eckard Modrow: Informatik mit Delphi 1/2 , Verlag emu online Bernard Schriek:Informatik mit Java /II, Nili-Verlag Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1992

Informatik, G-Kurs (vierstündig)	2. Halbjahr der Hauptphase
Datentypen und Datenstrukturen (Fortsetzung)	**30 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Baumstrukturen Bäume als spezielle Graphen; Wurzelbäume mit Wurzel, Knoten, Kanten, Blättern Begrifflichkeiten wie Ordnung, Höhe, Pfad Rekursive Definition	Hierarchische Beziehungen zwischen Objekten der Problemwelt werden durch baumartige Strukturen modelliert: Verzeichnisstrukturen, Stammbäume, Aufrufschemata, Struktur von Termen Bei Bäumen beliebiger Ordnung genügt eine informelle Darstellung der strukturellen Eigenschaften am graphischen Modell.
Binäre Bäume Definition und Aufbau über Knoten Traversierungsstrategien: Postorder, Preorder, Inorder	Binäre Bäume sollen wegen ihrer Bedeutung detaillierter behandelt werden.
Geordnete binäre Bäume Speicherung von Schlüsseln in einem binären Baum Elementare Operationen: Suchen, Einfügen und Entfernen	Insbesondere der geordnete binäre Baum ist als Struktur zur Speicherung von Schlüsseln eine der wichtigsten Datenstrukturen. Der Aufbau der Datenstruktur mit Knoten, deren Verlinkung und die Wirkungsweise der elementaren Operationen sind ebenso wie Verwaltungsoperationen detailliert zu behandeln.
Rekursion auf dynamischen Datenstrukturen	Listen und Bäume sind rekursive Strukturen. Die Implementation der elementaren Operationen auf diesen Strukturen liefert überzeugende Beispiele für eine vorteilhafte rekursive Formulierung von Algorithmen.
	Literatur/Medien Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Eckard Modrow: Informatik mit Delphi 1/2 , Verlag emu online Bernard Schriek:Informatik mit Java /II, Nili-Verlag

Informatik, G-Kurs (vierstündig)	2. Halbjahr der Hauptphase
Datentypen und Datenstrukturen (Fortsetzung)	**30 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Abstrakte Datentypen (ADT) ADT als Idee der Modellierung neuer Datentypen unabhängig von den Details einer Programmiersprache.	Die Modellierung von Datenstrukturen kann unabhängig von den Eigenheiten einer Programmiersprache und speziellen Implementierungsdetails erfolgen. Es genügt eine Beschreibung, wie die Werte der neuen Struktur aufgebaut sind, wie erzeugende und zugreifende Operationen wirken und wie sich die Datenstruktur nach außen verhält.
Modellierung und Implementierung der abstrakten Datentypen • Lineare Liste • Stapel • Schlange • Binärer Suchbaum als Klassen.	Die drei zentralen Aspekte eines ADT, Spezifikation – Implementation – Anwendung, sollen beim Modellieren und Implementieren dieser Datentypen demonstriert werden.
Problemlösungen unter Verwendung von ADT	In objektorientierten Sprachen werden ADT als Klassen implementiert, wichtige Datentypen sind über Klassenbibliotheken direkt verfügbar. Zur Verwendung dieser Datenstrukturen in Programmen genügen die Kenntnis der Signaturen der Methoden und eine Beschreibung ihrer Funktionsweise.
	Literatur/Medien Baumann R.: Informatik für die Sekundarstufe II, Klett-Verlag Stuttgart 1992 Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1992

Informatik, G-Kurs (vierstündig)	2. Halbjahr der Hauptphase
Suchen und Sortieren	**10 Stunden**

Verbindliche Inhalte	Vorschläge und Hinweise
Suchen in Reihungen • sequenzielle Suche • binäre Suche in sortierten Reihungen **Sortieren von Reihungen** **Einfache Sortierverfahren** • Bubblesort, Insertionsort **Rekursive Sortierverfahren** • Mergesort	Such- und Sortieralgorithmen sind Standardalgorithmen der Informatik und sollen analysiert, implementiert und im Rahmen von Komplexitätsbetrachtungen bewertet werden. Sie bieten vielfältige Möglichkeiten des Vergleiches iterativer und rekursiver Formulierungen von Algorithmen. Sie liefern Beispiele für die anschließende Algorithmenanalyse und die Klassifizierung von Algorithmen. Die Abschätzung und der Vergleich des von der Problemgröße abhängigen Rechenzeitbedarfes kann als Vorbereitung der Algorithmenanalyse (siehe Lehrplan 3. Kurshalbjahr) thematisiert werden. Eine vergleichende Betrachtung von Algorithmen zum Suchen und Sortieren in Reihungen und Listen zeigt, dass der Aufwand zur Problemlösung auch von der Wahl der Datenstruktur abhängig ist. **Literatur/Medien** Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000 Eckard Modrow: Informatik mit Delphi 1/2 , Verlag emu online Bernard Schriek:Informatik mit Java /II, Nili-Verlag

Informatik, G-Kurs (vierstündig)	3. Halbjahr der Hauptphase
Algorithmenanalyse, Grenzen der Berechenbarkeit	**10 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Algorithmenanalyse, Effizienzbetrachtungen Asymptotische Abschätzung des Zeitaufwands eines Algorithmus abhängig von der Problemgröße O-Notation zur Angabe oberer Schranken für die Komplexität von Algorithmen Klassifizierung bekannter Algorithmen gemäß der O-Notation	Zur Lösung des Sortierproblems werden Algorithmen unterschiedlicher Effizienz betrachtet. Dies motiviert die Notwendigkeit der Abschätzung des Zeitbedarfes von Algorithmen vor einer Auswahl unter mehreren möglichen Problemlösungen. Die O-Notation ermöglicht eine einfache Klassifizierung von Algorithmen bezüglich ihrer Effizienz.
Praktische Grenzen der Berechenbarkeit Algorithmen mit exponentieller Zeitkomplexität	Am Beispiel des Rundreiseproblems (oder von Stundenplan- und Kopplungsproblemen) kann die Existenz von Problemen, zu deren Lösung nur Algorithmen mit exponentieller Zeitkomplexität bekannt sind, demonstriert werden. Dies führt zur Thematisierung und Begründung der Existenz von praktischen Grenzen für die Berechenbarkeit.
Theoretische Grenzen der Berechenbarkeit Existenz nicht berechenbarer Funktionen Die Unentscheidbarkeit des Halteproblems	Es soll verdeutlicht werden, dass die Existenz von Problemen, die algorithmisch nicht lösbar sind, bewiesen ist, also der Berechenbarkeit durch einen Computer prinzipielle Grenzen gesetzt sind. **Literatur/Medien:** Barth, Algorithmik für Einsteiger, Vieweg, 2003

Informatik, G-Kurs (vierstündig)	3. Halbjahr der Hauptphase
Automaten und formale Sprachen	**20 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Endliche Automaten mit und ohne Ausgabe	Endliche Automaten zur Modellierung von Abläufen und zur Spezifikation des Verhaltens von Maschinen können an Beispielen aus der Praxis (Getränkeautomat, Parkscheinautomat) erklärt werden. Diese und weitere Beispiele aus der Steuerungstechnik führen zur Einführung der endlichen Automaten mit Ausgabe.
Beschreibung der Funktionsweise durch Übergangstabellen, Ausgabetabellen (endl. Automat mit Ausgabe) und Übergangsgraphen	
Formale Definition endlicher Automaten als 5-Tupel	Der endliche Akzeptor ist ein wichtiger Sonderfall endlicher Automaten. An den Beispielen eines Erzeugers für Paritätsbits und eines Prüfers für Paritätsbits kann die Funktionsweise der endlichen Automaten mit Ausgabe und der endlichen Akzeptoren verdeutlicht werden.
Konstruktion endlicher Automaten mit / ohne Ausgabe zu vorgegebenen Problemstellungen	
Deterministische und nichtdeterministische endliche Automaten	Die Konstruktion endlicher Akzeptoren zu Problemstellungen wie der Suche vorgegebener Teilstrings in Zeichenketten (pattern-matching) führt zum Problem nichtdeterministischer Automaten.
Formale Sprachen	
Definition	Endliche Automaten, die Folgen von Zeichen über einem Zeichenalphabet akzeptieren, führen zur Einführung des Begriffs formale Sprache.
Beschreibung durch - Aufzählung der Worte - charakterisierende Eigenschaften der Worte	Im Gegensatz zur Menge der Worte einer natürlichen Sprache können viele formale Sprachen über die Beschreibung kennzeichnender Eigenschaften der Wörter oder über Bildungsregeln festgelegt werden.
Grammatiken	
Komponenten einer Grammatik: Terminalsymbole, Variablen, Startvariable, Produktionsregeln	Für viele formale Sprachen ist eine vollständige Beschreibung durch charakterisierende Eigenschaften der Wörter nicht möglich. Grammatiken sind Regelwerke, welche die Herleitung aller Wörter einer Sprache ermöglichen.
Beispiele von Grammatiken und der zugehörigen formalen Sprachen	Der Zusammenhang zwischen Grammatiken und Sprachen wird an einfachen Beispielen (regulärer und kontextfreier Sprachen) erarbeitet.
Wortprüfung durch Ableitung	Von wesentlicher Bedeutung ist hierbei die Klärung der Frage, ob eine vorgegebene Folge von Terminalsymbolen nach den Produktionsregeln einer Grammatik aus der Startvariablen abgeleitet werden kann.

Informatik, G-Kurs (vierstündig)	3. Halbjahr der Hauptphase
Automaten und formale Sprachen (Fortsetzung)	**20 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Reguläre Sprachen als Sprachen endlicher Automaten / Akzeptoren Beschreibung durch - reguläre Grammatiken - reguläre Ausdrücke	Die zur Sprache eines endlichen Akzeptors gehörenden Worte werden stets nach einfachen Regeln (Konkatenation, Auswahl, Iteration) aus dem zugrunde liegenden Alphabet gebildet. Diese Regeln finden sich sowohl in den Bildungsgesetzen regulärer Ausdrücke als auch in den Produktionsregeln regulärer Grammatiken wieder.
Äquivalenz regulärer Sprachen und endlicher Akzeptoren	Jede durch einen endlichen Automaten festgelegte Sprache kann durch einen regulären Ausdruck oder eine reguläre Grammatik beschrieben werden und umgekehrt.
Konstruktion endlicher Akzeptoren zu regulären Sprachen, die beschrieben sind durch - reguläre Grammatiken - reguläre Ausdrücke	Zu vorgegebenen regulären Ausdrücken oder regulären Grammatiken kann nach einem einfach auszuführenden Verfahren der zugehörige endliche Akzeptor konstruiert werden. Allerdings führt dieses Verfahren in vielen Fällen zu nichtdeterministischen endlichen Akzeptoren.
Teilmengenkonstruktion	Das Verfahren der Teilmengenkonstruktion ermöglicht die Entwicklung gleichwertiger deterministischer Akzeptoren.
Kontextfreie Sprachen Beispiele nichtregulärer Sprachen	Am Beispiel der Sprache $L = \{a^n b^n\}$ oder der Sprache der Palindrome werden die Grenzen endlicher Akzeptoren deutlich.
Kontextfreie Grammatiken und Sprachen Zusammenhang zwischen kontextfreien Sprachen und Programmiersprachen Syntaxdiagramme als graphisches Beschreibungsmittel	Kontextfreie Grammatiken ermöglichen eine einfache Beschreibung dieser und anderer nichtregulärer Sprachen. Am Beispiel der Syntax arithmetischer Ausdrücke lässt sich zeigen, dass Programmiersprachen eine kontextfreie Grammatik zugrunde liegt. Die Prüfung eines Computerprogramms auf syntaktische Korrektheit beantwortet die Frage, ob das Programm nach den Regeln der zugrunde liegenden Grammatik abgeleitet werden kann. Syntaxdiagramme ermöglichen eine übersichtliche Darstellung einer kontextfreien Grammatik. **Literatur** Eckart Modrow: Theoretische Informatik mit Delphi Rolf Socher: Theoretische Grundlagen der Informatik, fV Leipzig Kastens,Büning: Modellierung; Hanser Verlag Materialien zu Kara, Exorciser: Schweizer Bildungsserver swissedu.ch

Informatik, G-Kurs (vierstündig)	4. Halbjahr der Hauptphase
Funktionsweise von Computersystemen	**15 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Aufbau und Funktionsweise eines Rechnersystems Prozessor, Hauptspeicher, Busse, IO-Einheiten, Hintergrundspeicher, Peripherie	Schülerinnen und Schüler sollen den technischen Aufbau und das Zusammenspiel der Komponenten eines Computers kennen und verstehen. Dabei bietet sich die Demonstration an geeigneter Hardware an.
Von Neumann-Rechner **Architektur**: Prozessor mit Steuerwerk, Registern, ALU, Speicher mit abgelegtem Programm und Daten **Arbeitsweise:** Instruktionszyklus mit Fetch, Increment, Decode, Fetch operands, Execute Befehlsarten, Befehlsdekodierung, Adressierung	Die Darstellung des von–Neumann–Modells einer Rechnerarchitektur dient dem Verständnis des grundlegenden Aufbaus und der internen Funktionsweise realer Mikroprozessorsysteme. An einem Mikrocomputermodell soll die Funktionsweise eines nach dem von-Neumann-Modell aufgebauten Systems demonstriert und geübt werden. Das Schreiben einfacher Maschinenprogramme mit dem Befehlssatz der Modellmaschine und deren schrittweise Abarbeitung führen zu einem vertieften Verständnis der allgemeinen Funktionsweise und der maschineninternen Abläufe. Maschinenmodell und Befehlssatz einer einfachen Registermaschine werden einheitlich vorgegeben.
Modell einer Registermaschine mit einem vorgegebenen einfachen Befehlssatz Architektur des Modellrechners Syntax und Semantik des Befehlssatzes Programmierung auf Maschinenebene Einfache Assemblerprogramme	
Programmiersprachen, Übersetzung und Interpretation Maschinensprache, Assemblersprache, Hochsprache Prinzipielle Funktionsweise von Compilern und Interpretern	Zur Vereinfachung der Programmierung wurden Assembler- und Hochsprachen entwickelt. Die Grundideen der Assemblerprogrammierung werden bei der Besprechung der Registermaschine erklärt. Daran anschließend sollen die Vorteile der Programmierung in einer Hochsprache dargelegt und die prinzipielle Arbeitsweise von Übersetzern und Interpretern erklärt werden. Es genügt eine informelle Darstellung; die vertiefte Behandlung des Themas „Übersetzerbau“ ist nicht vorgesehen. **Literatur** Eckart Modrow: Technische Informatik mit Delphi Materialien zum didaktischen Computer (siehe Anlage) Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1998

Informatik, G-Kurs (vierstündig)	4. Halbjahr der Hauptphase
Kommunikation und Sicherheit in Rechnernetzen	**15 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Rechnernetze Das Client-Server-Modell LAN, WAN, Router IP-Adressen	Netzwerke arbeiten überwiegend nach dem Client-Server-Modell, wobei die Daten in Paketen übermittelt werden. An Beispielen kann die verbindungslose bzw. verbindungsorientierte Kommunikation erläutert werden. Unterschiedliche Netze sind durch Router gekoppelt, wobei IP-Adressen zur Unterscheidung der am Netzwerk beteiligten Geräte dienen.
Internet Dienste und Protokolle	Die Begriffe Dienst und Protokoll können am Beispiel des Mediums Telefonleitung geklärt werden. Es genügt eine abgrenzende Begriffserklärung der Dienste WWW, E-Mail und der zugehörigen Protokolle HTTP, SMTP und POP. Dazu gehört auch die Klärung der Begriffe HTML, URL, Domain, DNS.
Schichtenmodell zu TCP/IP	Das Schichtenmodell zu TCP/IP beschreibt den Datentransport zwischen Rechnern. Die Aufgabenverteilung zwischen Anwendungsschicht, Transportschicht, Internetschicht und Netzzugangsschicht kann anhand des Schichtenmodells verdeutlicht werden.
Rechtliche Aspekte Rechtliche Aspekte bei der Nutzung von Netzdiensten	Hier sollen Sicherheitsprobleme bei der Kommunikation thematisiert werden (z.B. Urheberrecht, Copyright, Haftung bei Links). **Literatur** Eckart Modrow: Technische Informatik mit Delphi, emu online

Informatik, G-Kurs (vierstündig)	4. Halbjahr der Hauptphase
Kommunikation und Sicherheit in Rechnernetzen (Fortsetzung)	**15 Stunden**

verbindliche Inhalte	Vorschläge und Hinweise
Kryptographische Grundbegriffe Klartext, Geheimtext, Schlüssel Substitutions-, Transpositionsalgorithmus Chiffrieralgorithmus, das Prinzip von Kerckhoffs	Wiederholung der in der Einführungsphase behandelten Begriffe und Konzepte.
Sicherheit Vertraulichkeit, Authentizität, Integrität	Es genügt die Gegenüberstellung und Besprechung von 'perfekter Sicherheit' (One-time-pad) und der schwächeren Variante der 'informationstheoretischen Sicherheit' an Beispielen. Die kryptographischen Sicherheitsziele Vertraulichkeit, Authentizität, Integrität können an Fallbeispielen erläutert werden.
Symmetrische und asymmetrische Verschlüsselungsverfahren Symmetrische Verschlüsselung Einwegfunktion, Trapdoor-Einwegfunktion Asymmetrische Verschlüsselung	Die Besprechung von Eigenschaften und Unterschiede von symmetrischen und Public-Key-Verfahren wird an einfachen Szenarien durchgeführt.
RSA-Algorithmus • Modulare Arithmetik • Schlüsselerzeugung, Verschlüsselung, Entschlüsselung, Korrektheit der Entschlüsselung, Trapdoor-Eigenschaft der Verschlüsselungsfunktion, Sicherheit des RSA-Algorithmus • Blockweise Chiffrierung • Der RSA-Algorithmus als Signaturverfahren	Die erforderlichen zahlentheoretischen Grundlagen (Modulare Arithmetik in $(Z^*_n ; \bullet)$) müssen nicht bewiesen werden. Es genügt eine zum Verständnis des Verfahrens ausreichende exemplarische Darstellung. Ergänzend sind auch Programmieraufgaben möglich (z.B. Potenzierung, erweiterter euklidischer Algorithmus). Einwegfunktionen bzw. Trapdoor-Einwegfunktionen spielen in der Kryptographie eine entscheidende Rolle: Das RSA-Verfahren ist ein bekanntes Beispiel für die Verschlüsselung durch eine Trapdoor-Einwegfunktion. Die praktische Einübung des Algorithmus sollte sich auf Zahlen mit kleinem Modul beschränken. Ergänzend können fertige Programme zur Verschlüsselung benutzt werden. Die Erzeugung einer Signatur (elektronische Unterschrift) und ihre Verifikation ist ein weiteres Beispiel für die Bedeutung des RSA-Verfahrens. **Literatur/Medien** Harald Scheid: Elemente der Arithmetik und Algebra; 2. Auflage, BI-Verlag, 1992 Albrecht Beutelspacher: Kryptologie, Vieweg 1993 Beutelspacher, Schwenk, Wolfenstetter: Moderne Verfahren der Kryptographie, Vieweg 1998 Schmeh: Kryptografie, dpunkt.verlag; 2001