

Gymnasiale Oberstufe Saar (GOS)

Lehrplan Informatik

G-Kurs (zweistündig)

Februar 2008

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

Vorbemerkung

1. und 2. Kurshalbjahr

Die zentralen Themen im Lehrplan für die Kursphase sind die Idee des Algorithmus, der Modularisierung von Programmen. Dazu wird an das Wissen aus der Einführungsphase angeknüpft und in einem ersten Abschnitt das strukturierte Vorgehen bei der Realisierung komplexer Algorithmen thematisiert.

Die Umsetzung der Modelle in Programme erfordert vertiefte Kenntnisse einer objektorientierten Programmiersprache. Bei der Gestaltung von Benutzeroberflächen sollte auch auf Aspekte der Benutzungsfreundlichkeit eingegangen werden.

Unabhängig von der Programmierung soll von Beginn an das fundamentale Konzept der Objektorientierung als Leitlinie bei der Problemlösung herausgestellt werden. Insbesondere werden die Strategien beim objektorientierten Entwurf und die Vorgehensweise bei der Implementierung objektorientierter Modelle in Anwendungsbeispielen aufgezeigt.

Die Themen "Algorithmenanalyse, Grenzen der Berechenbarkeit" schließen sich an das Kapitel OOP an. Mit Hilfe der O-Notation sollen dort verwendete Algorithmen charakterisiert und klassifiziert werden. Die Berechenbarkeitstheorie beschäftigt sich mit den grundsätzlichen Möglichkeiten und Grenzen der Algorithmisierbarkeit.

Die Fachbegriffe und Zusammenhänge sollten möglichst anschaulich eingeführt werden, vertiefte mathematische Formalismen und Methoden sind zu vermeiden. Soweit möglich, sind die theoretischen Konzepte mit konkreten Anwendungen zu verbinden.

2. und 3. Kurshalbjahr

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 Sprachen werden die praktische Bedeutung formaler Sprachen bei der Kommunikation zwischen Mensch und Maschine sowie ihre Beschreibungsmöglichkeiten erklärt.

Das von John von Neumann beschriebene Konzept beschreibt einen universellen Rechner und vermittelt einen Einblick in die Prinzipien, nach denen Computer aufgebaut sind.

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 von asymmetrischen Verfahren dargestellt und in realen Anwendungen aus der Praxis erprobt.

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.

Stoffverteilungsplan

G-Fach Informatik 1. Halbjahr der Hauptphase (2 Wochenstunden)	
Objektorientiertes Modellieren und Programmieren	20

G-Fach Informatik 2. Halbjahr der Hauptphase (2 Wochenstunden)	
Anwendungen der Objektorientierung	10
Algorithmenanalyse, Grenzen der Berechenbarkeit	10

G-Fach Informatik 3. Halbjahr der Hauptphase (2 Wochenstunden)	
Automaten und formale Sprachen	15

G-Fach Informatik 4. Halbjahr der Hauptphase (2 Wochenstunden)	
Funktionsweise von Computersystemen	5
Kommunikation und Sicherheit in Rechnernetzen	15

verbindliche Inhalte

Vorschläge und Hinweise

Objektorientierte Analyse

Unabhängig von der Programmierung wird die Möglichkeit der objektorientierten Analyse als eine Lösungsstrategie thematisiert. Ein Ausschnitt der realen Welt wird untersucht und unter Vernachlässigung der unwichtigen auf die bedeutsamen Merkmale reduziert. Zur Vermittlung einer Objektsichtweise bieten sich nichtinformatische Beispiele wie die Reduktion einer Fotografie auf charakteristische Elemente und Strukturen oder ein Produktkatalog einer Firma mit seiner Gruppierung von Waren an. Hier können alle wesentlichen Objekteigenschaften erarbeitet werden.

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 gemeinsamen Eigenschaften und Verhaltensweisen werden in der Klassendeklaration beschrieben.

Methoden

Einfache Datentypen
 Deklaration: Formale Parameter, Parameterlisten, Signaturen
 Aufruf: Varianten der Parameterübergabe
 Gültigkeitsbereiche von Variablen und Parametern

 Modularisierung und Strukturierung von komplexen Problemlösungen

Algorithmen zur Lösung eines Problems werden als Methoden formuliert. Nach der Implementierung genügen das Verständnis der Funktionsweise und die Kenntnis der Schnittstelle (Signatur) beim Einsatz in 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 Rekursionsablaufs durch Aufrufschemaschemata.

Vergleich äquivalenter rekursiver und iterativer Varianten von Algorithmen zur gleichen Problemlösung.

Objektorientiertes Modellieren und Programmieren

20 Stunden

verbindliche Inhalte

Vorschläge und Hinweise

Datenstrukturen

Statische Strukturen

Ein- und zweidimensionale Reihungen
(Felder, Arrays)

Grundlegende Eigenschaften dynamischer Datenstrukturen

Unterschied statische – dynamische Struktur;
Vor- und Nachteile im Vergleich
Speicherorganisation und Verwaltung
Zeigerkonzept (Referenz, Referenzvariable)

Anwenden von Klassenbibliotheken

Zusammengesetzte Datentypen ermöglichen die Zusammenfassung von Daten gleichen oder unterschiedlichen Typs.

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.

In objektorientierten Sprachen werden wichtige Anwendungen als Klassen implementiert. Sie sind über Klassenbibliotheken direkt verfügbar. Zur Verwendung dieser Strukturen in Programmen genügen die Kenntnis der Signaturen der Methoden und eine Beschreibung ihrer Funktionsweise. (ADT, Such- und Sortierverfahren, Grafikroutinen, ...)

verbindliche Inhalte

Vorschläge und Hinweise

Objektorientiertes Modellieren und Programmieren

Modellierung von Klassen und einfachen statischen Beziehungen

Modellierungsbeispiel aus einem Teilgebiet der angewandten Informatik

Beim objektorientierten Modellieren werden Klassen gleichartiger Objekte identifiziert und Beziehungen zwischen den Klassen erkannt; die Ergebnisse dieser objektorientierten Analyse werden graphisch veranschaulicht.

Ideen und Konzepte der objektorientierten Programmierung sollen bei der Bearbeitung kleiner, thematisch begrenzter Programmierprojekte demonstriert und eingeübt werden. Es sollten in diesem Kapitel geeignete Algorithmen betrachtet werden, die in der Algorithmenanalyse aufgegriffen werden können.

Beispiele:

Alle Gebiete der Informatik können entsprechende Anwendungsbeispiele liefern und Einblicke in diese ermöglichen.

- Algorithmen zu Graphen
- Computergraphik
- Bioinformatik
- Suchen und Sortieren
- Text-Bild-Adventure, ...

Literatur/Medien

Küchlin/Weber: Einführung in die Informatik, Springer-Verlag Berlin 2000
 Bertrand Meyer, Object-oriented Software Construction, Sams 1997
 Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1998
 Helmut Balzert, Lehrbuch Grundlagen der Informatik. Spektrum Ak. Verlag 1999
 Siegfried Spolwig: Objektorientierung im Informatikunterricht. Dümmler-Verlag 1997

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

An den in Kapitel 'Objektorientiertes Modellieren und Programmieren' behandelten Problemen werden Algorithmen unterschiedlicher Effizienz betrachtet. Dies motiviert die Notwendigkeit der Abschätzung des Zeitbedarfs 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

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.

Beispiel:
Die Unentscheidbarkeit des Halteproblems

Literatur/Medien:
Barth, Algorithmik für Einsteiger, Vieweg, 2003
U. Schöning: Theoretische Informatik – kurzgefasst. Spektrum Akademischer Verlag 2001

verbindliche Inhalte

Vorschläge und Hinweise

Endliche Automaten mit und ohne Ausgabe

Beschreibung der Funktionsweise durch Übergangstabellen, Ausgabetafeln (endliche Automat mit Ausgabe) und Übergangsgraphen

Formale Definition endlicher Automaten als 5-Tupel

Konstruktion endlicher Automaten mit / ohne Ausgabe zu vorgegebenen Problemstellungen

Deterministische und nichtdeterministische endliche Automaten

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.

Der endliche Akzeptor ist ein wichtiger Sonderfall endlicher Automaten. An den Beispielen eines Erzeugers für Paritätsbits und eines Prüfers für Paritätsbits kann die Funktionsweise der endlichen Automaten mit Ausgabe und der endlichen Akzeptoren verdeutlicht werden.

Die Konstruktion endlicher Akzeptoren zu Problemstellungen wie der Suche vorgegebener Teilstrings in Zeichenketten (pattern-matching) führt zum Problem nichtdeterministischer Automaten.

Formale Sprachen

Definition

Beschreibung durch
 - Aufzählung der Worte
 - charakterisierende Eigenschaften der Worte

Endliche Automaten, die Folgen von Zeichen über einem Zeichenalphabet akzeptieren, führen zur Einführung des Begriffs formale Sprache.

Im Gegensatz zur Menge der Worte einer natürlichen Sprache können viele formale Sprachen über die Beschreibung kennzeichnender Eigenschaften der Wörter oder über Bildungsregeln festgelegt werden.

Grammatiken

Komponenten einer Grammatik: Terminalsymbole, Variablen, Startvariable, Produktionsregeln

Beispiele von Grammatiken und der zugehörigen formalen Sprachen

Wortprüfung durch Ableitung

Für viele formale Sprachen ist eine vollständige Beschreibung durch charakterisierende Eigenschaften der Wörter nicht möglich. Grammatiken sind Regelwerke, welche die Herleitung aller Wörter einer Sprache ermöglichen.

Der Zusammenhang zwischen Grammatiken und Sprachen wird an einfachen Beispielen (regulärer und kontextfreier Sprachen) erarbeitet.

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.

verbindliche Inhalte

Vorschläge und Hinweise

Reguläre Sprachen als Sprachen endlicher Automaten / Akzeptoren

Beschreibung durch

- reguläre Grammatiken
- reguläre Ausdrücke

Äquivalenz regulärer Sprachen und endlicher Akzeptoren

Konstruktion endlicher Akzeptoren zu regulären Sprachen, die beschrieben sind durch

- reguläre Grammatiken
- reguläre Ausdrücke

Beispiele nichtregulärer Sprachen

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.

Jede durch einen endlichen Automaten festgelegte Sprache kann durch einen regulären Ausdruck oder eine reguläre Grammatik beschrieben werden und umgekehrt.

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 nicht-deterministischen endlichen Akzeptoren.

Am Beispiel der Sprache $L = \{a^n b^n\}$ oder der Sprache der Palindrome werden die Grenzen endlicher Akzeptoren deutlich.

Literatur

Rolf Socher: Theoretische Grundlagen der Informatik, fV Leipzig
 Kastens, Büning: Modellierung; Hanser Verlag
 Materialien zu Kara, Exorciser: Schweizer Bildungsserver swissedu.ch

Funktionsweise von Computersystemen

5 Stunden

verbindliche Inhalte

Vorschläge und Hinweise

Aufbau und Funktionsweise eines Rechnersystems

Prozessor, Hauptspeicher, Busse, IO-Einheiten, Hintergrundspeicher, Peripherie

Von Neumann-Prinzipien

Architektur:

Prozessor mit Steuerwerk, Registern, ALU, Speicher mit abgelegtem Programm und Daten

Arbeitsweise:

Instruktionszyklus mit Fetch, Increment, Decode, Fetch operands, Execute Befehlsarten, Befehlsdekodierung, Adressierung

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.

Zusammenstellung der Strukturen und Arbeitsweisen nach denen der von-Neumann-Rechner Programme abarbeitet

Die Darstellung des von-Neumann-Modells einer Rechnerarchitektur dient dem Verständnis des grundlegenden Aufbaus und der internen Funktionsweise realer Mikroprozessorsysteme.

Literatur

Rollke/Sennholz: Grund- und Leistungskurs Informatik, Cornelsen-Verlag 1998

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

Douglas Comer, *Computernetzwerke und Internets*, Pearson Studium, 2002

verbindliche Inhalte

Vorschläge und Hinweise

Kryptographische Grundbegriffe

Klartext, Geheimtext, Schlüssel
 Substitutions-, Transpositionsalgorithmus
 Chiffrieralgorithmus, das Prinzip von
 Kerckhoffs

Wiederholung der in der Einführungsphase be-
 handelten Begriffe und Konzepte.

Sicherheit

Vertraulichkeit, Authentizität, Integrität

Es genügt die Gegenüberstellung und Bespre-
 chung von 'perfekter Sicherheit' (One-time-pad)
 und der schwächeren Variante der 'informati-
 onstheoretischen Sicherheit' an Beispielen.
 Die kryptographischen Sicherheitsziele Vertrau-
 lichkeit, Authentizität, Integrität können an Fall-
 beispielen erläutert werden.

**Symmetrische und asymmetrische
 Verschlüsselungsverfahren**

Symmetrische Verschlüsselung
 Einwegfunktion, Trapdoor-Einwegfunktion
 Asymmetrische Verschlüsselung

Die Besprechung von Eigenschaften und Unter-
 schieden von symmetrischen und Public-Key-
 Verfahren wird an einfachen Szenarien durch-
 geführt.

Einwegfunktionen bzw. Trapdoor-Einwegfunktio-
 nen spielen in der Kryptographie eine entschei-
 dende Rolle: Das RSA-Verfahren ist ein be-
 kanntes Beispiel für die Verschlüsselung durch
 eine Trapdoor-Einwegfunktion, das in vielen
 Anwendungsgebieten zum Einsatz kommt.

**Anwendung von standardisierten
 Verschlüsselungs- und Signaturverfahren**

Die Behandlung ist möglich mit dem Standard
OpenPGP. Auch die Erzeugung einer Signatur
 (elektronische Unterschrift) und ihre Verifikation,
 lassen sich am Beispiel von PGP verdeutlichen.

Literatur/Medien

Albrecht Beutelspacher: Kryptologie, Vieweg
 1993
 Beutelspacher, Schwenk, Wolfenstetter: Moderne
 Verfahren der Kryptographie, Vieweg 1998
 Schmech: Kryptografie, dpunkt.verlag; 2001