18. Informatik

A. Fachbezogene Hinweise

Die Rahmenrichtlinien Informatik sind so offen formuliert, dass sie Raum für die Gestaltung eines zeitgemäßen Informatikunterrichts lassen.
Neue Inhalte der Informatik lassen sich unter die vorgegebenen Unterrichtsinhalte subsumieren. So findet sich in den Rahmenrichtlinien (RRL) z. B. zwar nicht der Begriff „Internet“, ein Informatikunterricht, in dem das Internet nicht an geeigneten Stellen thematisch Niederschlag findet, ist heute jedoch kaum vorstellbar.

Für die Thematischen Schwerpunkte des Zentralabiturs ergeben sich deshalb die folgenden Konsequenzen:

- Die für die Abiturprüfung verpflichtenden Kerninhalte der RRL und der Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA) Informatik bilden die Grundlage für die Aufgabenstellungen des Zentralabiturs.
- Zeitgemäße Abituraufgaben können sich nicht auf in den RRL explizit genannte Inhalte beschränken (vgl. „Internet“).
- Die vorliegenden Thematischen Schwerpunkte beschreiben den stofflichen Umfang der Aufgaben des Zentralabiturs 2016. Sie sollen die Inhalte eines zeitgemäßen Informatikunterrichts widerspiegeln, sind aber nicht so angelegt, dass dadurch die in der Qualifikationsphase zur Verfügung stehende Unterrichtszeit vollständig ausgefüllt wird.

Für Unterricht auf erhöhtem Anforderungsniveau werden in den jeweiligen Themenbereichen Ergänzungen angegeben, die zusätzlich zu den genannten Themen zu behandeln sind.

Reihenfolge der Thematischen Schwerpunkte:
Die beiden ersten Thematischen Schwerpunkte sind im ersten Schuljahrgang der Qualifikationsphase zu unterrichten. Der Thematische Schwerpunkt 3 wird für die Abiturprüfung 2017 als Thematischer Schwerpunkt 1 übernommen.

B. Thematische Schwerpunkte

Thematischer Schwerpunkt 1: Anwendung von Hard- und Softwaresystemen sowie deren gesellschaftliche Auswirkungen

Chiffrieren und Codieren

- Kryptografische Verfahren
  - monoalphabetische Verfahren (u.a. Caesar), polyalphabetische Verfahren (u.a. Vigenère)
  - Analyse monoalphabetischer Verfahren (Häufigkeitsanalyse)
  - Implementierung klassischer Verschlüsselungsverfahren

- Codierung
  - Anwenden und Analysieren eines gegebenen verlustfreien Codierungsverfahrens
  - komprimierende Codes (Lauflängencodierung, Huffman-Codierung (ohne Implementierung))

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- fehlererkennende und fehlerkorrigierende Codes (u.a. Paritätsbit, (7,4)-Hamming-Code)

Datenschutz und Datensicherheit

- Einsatz von asymmetrischen Verfahren zur Authentifikation (allgemeines Prinzip, digitale Signatur)
- Erläuterung grundlegender Begriffe im Kontext der informationellen Selbstbestimmung

Thematischer Schwerpunkt 2: Werkzeuge und Methoden der Informatik

Algorithmen (auch rekursive)

- Erstellung eines Algorithmus zu einem gegebenen Problem in schriftlich verbalisierter Form oder als Struktogramm
- Bearbeitung eines Algorithmus, gegeben durch ein Struktogramm, Pseudocode oder Code in Wortform
  - Analyse, u.a. mit einer Tracetabelle, durch Auswahl geeigneter Testdaten
  - Vervollständigung
  - Präzisierung
  - Korrektur
- Implementierung eines Algorithmus in Java oder einer vergleichbaren Programmiersprache

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Abschätzen der Komplexität eines Algorithmus

Objektorientierte Modellierung

- Klassendiagramme (Vererbung, Aggregation, Assoziation)
- Anwendung der Klassen (ADTs) „Schlange“ und „Stapel“

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Implementierung der oben genannten Klassen
- Anwendung der Klasse (ADT) „Binärbaum“

Thematischer Schwerpunkt 3: Funktionsprinzipien von Hard- und Softwaresystemen einschließlich theoretischer bzw. technischer Modellvorstellungen

Endliche Automaten

- Analyse und Synthese von deterministischen und nichtdeterministischen endlichen Automaten und Mealy-Maschinen
  - Entwicklung eines Zustandsgraphen für ein gegebenes Problem
  - Analyse eines gegebenen Zustandsgraphen
  - Erweiterung eines gegebenen Zustandsgraphen

Reguläre Sprachen

- Analyse einer gegebenen Sprache
- Entwicklung einer Sprache für ein gegebenes Problem
- Umsetzung eines Zustandsgraphen in eine Grammatik und umgekehrt
- Syntaxdiagramme

Ergänzung für Unterricht auf erhöhtem Anforderungsniveau

- kontextfreie Sprachen
  - Analyse und Entwicklung einer Sprache
  - Ableitungsbäume und Syntaxdiagramme

C. Sonstige Hinweise

- Die Aufgabentexte selber enthalten keinen Code in einer konkreten Programmiersprache. Diejenigen Aufgabenteile, die die Implementierung in einer konkreten Programmiersprache erfordern, sind von den Schülerinnen und Schülern in Java oder einer vergleichbaren objektorientierten Programmiersprache zu bearbeiten.
- Anstelle der unterschiedlichen, sprachspezifischen Bezeichnungen „Prozedur“, „Funktion“ bzw. „Methode“ wird in den Aufgabenstellungen der Begriff „Operation“ verwendet.
- Die Ausgabe der Mealy-Maschine verwendet gegebenenfalls auch aus dem Ausgabealphabet gebildete Wörter.
- Aufgaben, die am Rechner zu bearbeiten sind, werden nicht gestellt.
- Das Lehrermaterial wird weiterhin ausführliche Lösungsskizzen enthalten. Die Implementierungen in einer Programmiersprache werden nur in Java vorgelegt.
- Die Anlage (Operationen der Klassen Stapel, Schlange und Binärbaum) ist als Hilfsmittel in der Abiturprüfung zugelassen.

Anlage

Operationen der Klassen Stapel, Schlange und Binärbaum

Die Klassen benutzen Inhaltsklassen bzw. -typen, die jeweils der aktuellen Aufgabenstellung angepasst werden. Die Klassen werden in den Aufgabenstellungen gegebenenfalls um Attribute und weitere Operationen ergänzt.
Die im Folgenden benutzte Notation entspricht der UML-Notation in Klassendiagrammen.
Mögliche Laufzeitfehler bei der Anwendung der Operationen, z. B. „entnehmen“ bei einem leeren Stapel, müssen bei der Bearbeitung entsprechender Aufgaben explizit abgefangen werden.

Stapel

`Stapel()`
Ein leerer Stapel wird angelegt.

`istLeer(): Wahrheitswert`
Wenn der Stapel kein Element enthält, wird der Wert *wahr* zurückgegeben, sonst der Wert *falsch*.

`inhaltGeben(): Inhaltstyp`
Der Inhalt des obersten Elements des Stapels wird zurückgegeben, das Element aber nicht entfernt.

`ablegen(Inhaltstyp inhalt)`
Ein neues Element mit dem angegebenen Inhalt wird auf den Stapel gelegt.

`entnehmen(): Inhaltstyp`
Der Inhalt des obersten Elements wird zurückgegeben und das Element wird entfernt.

Schlange

`Schlange()`
Eine leere Schlange wird angelegt.

`istLeer(): Wahrheitswert`
Wenn die Schlange kein Element enthält, wird der Wert *wahr* zurückgegeben, sonst der Wert *falsch*.

`inhaltGeben(): Inhaltstyp`
Der Inhalt des ersten Elements der Schlange wird zurückgegeben, das Element aber nicht entfernt.

`anhaengen(Inhaltstyp inhalt)`
Ein neues Element mit dem angegebenen Inhalt wird angelegt und am Ende an die Schlange angehängt.

`entnehmen(): Inhaltstyp`
Der Inhalt des ersten Elements wird zurückgegeben und das Element wird entfernt.

Binärbaum

`Baum()`
Ein leerer Baum wird erzeugt.

`Baum(Inhaltstyp inhalt)`
Ein Baum wird erzeugt. Die Wurzel erhält den übergebenen Inhalt als Wert.

`istLeer(): Wahrheitswert`
Die Anfrage liefert den Wert *wahr*, wenn der Baum leer ist, sonst liefert sie den Wert *falsch*.

`istBlatt(): Wahrheitswert`
Die Anfrage liefert den Wert *wahr*, wenn der Baum keine Nachfolger hat, sonst liefert sie den Wert *falsch*.

`linkerTeilbaum(): Baum`
Die Operation gibt den linken Teilbaum zurück. Existiert kein linker Nachfolger, so ist das Ergebnis *null*.

`rechterTeilbaum(): Baum`
Die Operation gibt den rechten Teilbaum zurück. Existiert kein rechter Nachfolger, so ist das Ergebnis *null*.

`linkenTeilbaumSetzen(Baum b)`
Der übergebene Baum wird als linker Teilbaum gesetzt.

`rechtenTeilbaumSetzen(Baum b)`
Der übergebene Baum wird als rechter Teilbaum gesetzt.

`inhaltGeben(): Inhaltstyp`
Die Operation gibt den Inhaltswert der Wurzel des aktuellen Baumes zurück.

`inhaltSetzen(Inhaltstyp inhalt)`
Die Operation setzt den Inhaltswert der Wurzel des aktuellen Baumes.