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 Informatik (EPA) 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 2018. 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 ist anschließend zu unterrichten. Er wird für die Abiturprüfung 2019 als Thematischer Schwerpunkt 1 übernommen.

B. Thematische Schwerpunkte

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

Datenbanken

- ER-Modell
  - Entwicklung und Erweiterung eines ER-Modells für ein vorgegebenes System
  - Analyse eines vorgegebenen ER-Modells bezüglich eines Anwendungsfalls
  - Umsetzung eines ER-Modells in ein relationales Datenbankschema
  - Analyse einer Tabellenstruktur bezüglich der ersten, zweiten und dritten Normalform
  - Entwicklung eines relationalen Datenbankschemas unter Berücksichtigung der ersten, zweiten und dritten Normalform
- SQL
  - Beschreibung der Wirkungsweise grundlegender SQL-Abfragen zur Datenbankauswertung anhand eines konkreten Satzes von Relationen
  - Entwurf und Anwendung einfacher Abfragen und Verbundabfragen in SQL

  Ergänzung für Kurse auf erhöhtem Anforderungsniveau
  - Entwurf und Anwendung geschachtelter SQL- Abfragen

Datenschutz und Urheberrecht

- Erläuterung grundlegender Begriffe im Kontext der informationellen Selbstbestimmung
- Diskussion rechtlicher Aspekte bei der Speicherung, Auswertung und Veröffentlichung von Daten

Thematischer Schwerpunkt 2: Werkzeuge und Methoden der Informatik

Algorithmen (auch rekursive)

- Erstellung eines Algorithmus in schriftlich verbalisierter Form und als Struktogramm
- Bearbeitung eines Algorithmus, gegeben durch ein Struktogramm oder in schriftlich verbalisierter Form
  - 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“, „Stapel“ und „Dynamische Reihung"

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Implementierung und Anwendung der Klasse (ADT) „Binärbaum“

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

Schaltnetze

- Entwicklung eines Schaltnetzes (Schaltwerttabelle, Schaltfunktionen, Gatterdarstellung) u.a. Halbaddierer, Volladdierer, Multiplexer
- Analyse einer vorgegebenen Gatterdarstellung
- Systematische Vereinfachung von Schalttermen
- Entwicklung eines Schaltnetzes unter Verwendung vorgegebener Schaltnetzkomponenten u.a. Paralleladdierer aus Halb- und Volladdierern

Endliche Automaten mit und ohne Ausgabe

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

Ergänzung für Unterricht auf erhöhtem Anforderungsniveau
Schaltwerke (ohne inneren Aufbau von Speicherbausteinen)

- Entwicklung eines Schaltwerkes mit vorgegebenen Eigenschaften u.a. Zähler, Schieberegister
- Analyse des Verhaltens eines gegebenen taktgesteuerten Systems auch unter Verwendung eines Zeitliniendiagramms
- Entwicklung eines Schaltwerkes aus einem Zustandsgraphen

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 Programmiersprache zu bearbeiten.
- Anstelle der unterschiedlichen, sprachspezifischen Bezeichnungen „Prozedur“, „Funktion“ bzw. „Methode“ wird in den Aufgabenstellungen der Begriff „Operation“ verwendet.
- Zur Vereinfachung sind beim Mealy-Automaten bei einem Zustandsübergang statt eines einzelnen Ausgabezeichens auch Wörter zulässig, die aus dem Ausgabealphabet gebildet werden.
- 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 dynamische Reihung, Stapel, Schlange und Binärbaum; Notation der SQL-Klauseln) ist als Hilfsmittel in der Abiturprüfung zugelassen.

Anlage

1. Operationen der Klassen dynamische Reihung, Schlange, Stapel 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.

Dynamische Reihung

Die Nummerierung der dynamischen Reihung beginnt mit dem Index 1.

`DynArray()`

Eine leere dynamische Reihung wird angelegt.

`isEmpty(): Wahrheitswert`

Wenn die Reihung kein Element enthält, wird der Wert *wahr* zurückgegeben, sonst der Wert *falsch*.

`getItem(Ganzzahl index): Inhaltstyp`

Der Inhaltswert des Elements an der Position `index` wird zurückgegeben.

`getLength(): Ganzzahl`

Die Anzahl der Elemente der dynamischen Reihung wird zurückgegeben.

`append(Inhaltstyp inhalt)`

Ein neues Element mit dem angegebenen Inhaltswert wird am Ende der dynamischen Reihung eingefügt.

`insertAt(Ganzzahl index, Inhaltstyp inhalt)`

Ein neues Element mit dem angegebenen Inhaltswert wird an der Position `index` in die dynamische Reihung eingefügt, falls an dieser Position schon ein Element in der Reihung vorhanden ist. Das sich vorher an dieser Position befindende Element und alle weiteren werden nach hinten verschoben.

Ist der Wert von `index` um eins größer als die Anzahl der Elemente der Reihung, wird der Inhaltswert wie bei `append` am Ende der dynamischen Reihung eingefügt.

Für alle weiteren Werte von `index` hat die Operation keine Wirkung.

`setItem(Ganzzahl index, Inhaltstyp inhalt)`

Der Inhaltswert des Elements an der Position `index` wird durch `inhalt` ersetzt. Falls die angegebene Position nicht existiert, hat die Operation keine Wirkung.

`delete(Ganzzahl index)`

Das Element an der Position `index` wird entfernt. Alle folgenden Elemente werden um eine Position nach vorne geschoben. Falls die angegebene Position nicht existiert, hat die Operation keine Wirkung.

Schlange

`Queue()`

Eine leere Schlange wird angelegt.

`isEmpty(): Wahrheitswert`

Wenn die Schlange kein Element enthält, wird der Wert *wahr* zurückgegeben, sonst der Wert *falsch*.

`head(): Inhaltstyp`

Der Inhaltswert des ersten Elements der Schlange wird zurückgegeben, das Element aber nicht entfernt.

`enqueue(Inhaltstyp inhalt)`

Ein neues Element mit dem angegebenen Inhaltswert wird angelegt und am Ende an die Schlange angehängt.

`dequeue(): Inhaltstyp`

Der Inhaltswert des ersten Elements wird zurückgegeben und das Element wird entfernt.

Stapel

`Stack()`
Ein leerer Stapel wird angelegt.
`isEmpty(): Wahrheitswert`
Wenn der Stapel kein Element enthält, wird der Wert *wahr* zurückgegeben, sonst der Wert *falsch*.
`top(): Inhaltstyp`
Der Inhaltswert des obersten Elements des Stapels wird zurückgegeben, das Element aber nicht entfernt.
`push(Inhaltstyp inhalt)`
Ein neues Element mit dem angegebenen Inhaltswert wird auf den Stapel gelegt.
`pop(): Inhaltstyp`
Der Inhaltswert des obersten Elements wird zurückgegeben und das Element wird entfernt.

Binärbaum

`BinTree(Inhaltstyp inhalt)`
Ein Baum wird erzeugt. Die Wurzel erhält den übergebenen Inhalt als Wert.
`isLeaf(): Wahrheitswert`
Die Operation liefert den Wert *wahr*, wenn der Baum keine Teilbäume besitzt, sonst liefert sie den Wert *falsch*.
`getLeft(): BinTree`
Die Operation gibt den linken Teilbaum zurück. Existiert kein linker Teilbaum, so ist das Ergebnis *null*.
`getRight(): BinTree`
Die Operation gibt den rechten Teilbaum zurück. Existiert kein rechter Teilbaum, so ist das Ergebnis *null*.
`setLeft(BinTree b)`
Der übergebene Baum wird als linker Teilbaum gesetzt.
`setRight(BinTree b)`
Der übergebene Baum wird als rechter Teilbaum gesetzt.
`getItem(): Inhaltstyp`
Die Operation gibt den Inhaltswert der Wurzel des aktuellen Baumes zurück.
`setItem(Inhaltstyp inhalt)`
Die Operation setzt den Inhaltswert der Wurzel des aktuellen Baumes.

2. SQL-Klauseln

SELECT-Anweisung

```
SELECT [ALL|DISTINCT] Select-Ausdruck {,Select-Ausdruck}
  FROM TabellenName {,TabellenName}
  [WHERE Bedingung]
  [GROUP BY SpaltenName {,SpaltenName}]
  [HAVING Bedingung]
  [ORDER BY Select-Ausdruck [ASC|DESC]{,Select-Ausdruck [ASC|DESC]}]
  [LIMIT Zeilenzahl]
```

Ein Select-Ausdruck kann ein Spaltenname, eine Spaltenfunktion oder ein Alias-Name sein.

Operatoren

+, -, *, /,
=, != (ungleich), >, <, >=, <=,
AND, OR, NOT,
LIKE, BETWEEN, IN, IS NULL

Arithmetische Gruppenfunktionen

AVG( ), COUNT( ), MAX( ), MIN( ), SUM( )