

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 Aufgabensstellungen 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 2014. 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 2015 als Thematischer Schwerpunkt 1 übernommen.

B. Thematische Schwerpunkte

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

Datenbanken

1. E-R-Modell

- Entwicklung eines E-R-Modells für ein vorgegebenes System
- Analyse eines vorgegebenen E-R-Modells bezüglich eines Anwendungsfalls
- Erweiterung eines vorgegebenen E-R-Modells
- Umsetzung eines E-R-Modells in ein relationales Datenbankmodell
- Normalisierung

2. SQL

- Beschreibung der Wirkungsweise grundlegender SQL-Abfragen zur Datenbankauswertung anhand eines konkreten Satzes von Relationen
- Einfache Abfragen und Verbundabfragen in SQL entwerfen und anwenden

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Geschachtelte SQL- Abfragen

Datenschutz und Urheberrecht

- Erläuterung grundlegender Begriffe im Kontext der informationellen Selbstbestimmung
- Rechtliche Aspekte bei der Veröffentlichung von Daten im Internet
- Beurteilung eines Anwendungsfalls

Thematischer Schwerpunkt 2: Werkzeuge und Methoden der Informatik*Algorithmen (allgemein)*

- 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, z. B. mit einer Durchlauftabelle (Trace) oder 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) „Liste“, „Schlange“ und „Stapel“

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Implementierung der oben genannten Klassen
- Graphen
Adjazenzmatrizen und Adjazenzlisten
Tiefen- und Breitensuche

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

- Entwicklung eines Schaltnetzes mit vorgegebenen Eigenschaften (Schaltwerttabelle, Schaltfunktionen, Gatterdarstellung)
- Analyse einer vorgegebenen Gatterdarstellung
- Entwicklung einer Schaltung mit vorgegebenen Eigenschaften aus gegebenen Komponenten
- Systematische Vereinfachung von Schalttermen

Endliche Automaten

- Entwicklung eines Zustandsgraphen für ein gegebenes Problem
- Analyse eines gegebenen Zustandsgraphen
- Erweiterung eines gegebenen Zustandsgraphen
- Umsetzung eines endlichen Automaten in ein Schaltwerk

Ergänzung für Kurse auf erhöhtem Anforderungsniveau

- Turingmaschinen
Entwicklung einer Turingmaschine für ein gegebenes Problem
Analyse einer gegebenen Turingmaschine
Erweiterung einer gegebenen Turingmaschine

C. Sonstige Hinweise

- Die Aufgabentexte selber enthalten keinen Code in einer konkreten Programmiersprache. Diejenigen Aufgabenteile, die die Implementation 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. Der Begriff „Klasse“ umfasst auch den den Begriff „Abstrakter Datentyp“.
- Aufgaben, die am Rechner zu bearbeiten sind, werden nicht gestellt.
- Taschenrechner, grafikfähige Taschenrechner oder CAS-Rechner sind als Hilfsmittel nicht zugelassen.
- Das Lehrermaterial wird weiterhin ausführliche Lösungsskizzen enthalten. Die Implementationen in einer Programmiersprache werden nur in Java vorgelegt.

Die Anlagen (Operationen der Klassen Liste, Stapel und Schlange, Notation der Turingmaschinen und SQL-Klauseln) sind als Hilfsmittel in der Abiturprüfung zugelassen.

Anlagen

1. Operationen der Klassen Liste, Stapel und Schlange

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.

Liste

Liste()

Eine leere Liste wird angelegt. Der interne Positionszeiger, der das aktuelle Element markiert, wird auf Null gesetzt.

istLeer(): Wahrheitswert

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

inhaltGeben(): Inhalt

Der Inhalt des aktuellen Listenelements wird zurückgegeben.

einfuegen(Inhalt inhalt)

Ein neues Listenelement mit dem angegebenen Inhalt wird angelegt und hinter der aktuellen Position in die Liste eingefügt, alle weiteren Elemente werden nach hinten verschoben. Der interne Positionszeiger steht auf dem neu eingefügten Element.

einfuegen(Ganzzahl position, Inhalt inhalt)

Ein neues Listenelement mit dem angegebenen Inhalt wird angelegt und an der angegebenen Position in die Liste eingefügt. Das vorher an dieser Position befindliche Element und alle weiteren Elemente werden nach hinten verschoben. Der interne Positionszeiger steht auf dem neu eingefügten Element. Falls die angegebene Position nicht existiert, hat die Operation keine Wirkung.

loeschen()

Das aktuelle Listenelement wird gelöscht. Der interne Positionszeiger steht anschließend auf dem Nachfolger des gelöschten Elements. Falls kein Nachfolger existiert, zeigt er auf den Vorgänger.

positionGeben(): Ganzzahl

Der Wert des internen Positionszeigers wird zurückgegeben.

positionSetzen(Ganzzahl position)

Der interne Positionszeiger wird auf den angegebenen Wert gesetzt. Falls die angegebene Position nicht existiert, wird der Positionszeiger nicht verändert.

laengeGeben(): Ganzzahl

Die Anzahl der Elemente in der Liste wird zurückgegeben.

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(): Inhalt

Der Inhalt des obersten Elements des Stapels wird zurückgegeben, das Element aber nicht entfernt.

ablegen(Inhalt inhalt)

Ein neues Element mit dem angegebenen Inhalt wird auf den Stapel gelegt.

entnehmen(): Inhalt

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(): Inhalt

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

anhaengen(Inhalt inhalt)

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

entnehmen(): Inhalt

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

2. Notation der Turingmaschinen

Bandalphabet = { *, 1 }

Ein leeres Band enthält nur Sterne.

Die Position des Kopfes der Turingmaschine wird durch einen Unterstrich gekennzeichnet, z. B.

symbolisiert *****111*****, dass der Kopf unter der rechten 1 des Eingabewortes steht.

Die Maschine wird durch einen Zustandsgraphen beschrieben.

Der Anfangszustand wird durch einen Pfeil gekennzeichnet, der Endzustand durch eine doppelte Umrandung.

An den Kanten stehen die Informationen in der folgenden Reihenfolge:

gelesenes Zeichen, geschriebenes Zeichen, Kopfbewegung.

Kopfbewegung: L: nach links, R: nach rechts, N: keine Bewegung.

Im Zustandsdiagramm nicht aufgeführte Übergänge führen zu einem Fehlerzustand.

3. SQL-Klauseln

SELECT-Anweisung

```
SELECT [ALL|DISTINCT] Select-Ausdruck {,Select-Ausdruck}
  FROM TabellenName {,TabelleName}
  [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( )
```