



Algorithmen und Datenstrukturen

**Python und PHP parallel · Anschluss an das
Grundlagentext**

**Datenstrukturen wählen · Algorithmen umsetzen · Ergebnisse
prüfen**

BG Abiturkern · BKWI1 Vertiefung

Didaktisch erweiterte Kursfassung · A5 · Schuljahr 2026/2027 · v1.5 · Serienmaster
R7

**Christine Janischek · Lizenz Custom License NC School-Only 1.0 ·
emotionalspirit.de**

*Dieses Lernskript wurde mit Unterstützung einer KI erstellt und anschließend redaktionell geprüft.
Trotz sorgfältiger Bearbeitung können Fehler enthalten sein. Bitte melde Fehler und
Verbesserungsvorschläge über die angegebene Kontaktmöglichkeit.*

Inhaltsverzeichnis

Die Einträge führen direkt zu den Kapiteln und Hauptabschnitten. In
den Überschriften bringt dich „↩ Inhalt“ hierher zurück.

- 01 Datenstrukturen im Überblick**
- 02 Operationen vergleichen und auswählen**
- 03 Warteschlange praktisch umsetzen**
- 04 Sortieren: Selection Sort und Bubble Sort**

05 Suchen: lineare und binäre Suche**06 Persistenz: JSON und relationale Datenbank****Den eigenen Lernweg sichtbar machen****WARUM WIR DEN DENKWEG FESTHALTEN**

Nicht nur die richtige Lösung ist wertvoll. Auch eine zunächst falsche Vermutung, eine hilfreiche Rückfrage und die eigene Korrektur zeigen, wie Lernen stattgefunden hat. Das Protokoll hilft dir bei der Selbstreflexion und ermöglicht der Lerngruppe, typische Denkwege und Irrwege später gemeinsam zu untersuchen.

Eintrag	Das notierst du knapp in eigenen Worten
1 · Aufgabe und Ziel	Was sollte ich herausfinden, darstellen oder programmieren?
2 · Ausgangsidee	Wie wollte ich beginnen? Welche Annahme hatte ich?
3 · Konkrete Hürde	An welcher Stelle kam ich nicht weiter?
4 · Mein Denk-Prompt	Welche Frage habe ich dem Lerncoach gestellt?
5 · Erhaltene Hilfe	Nur die entscheidende Denkfrage oder Hinweisstufe – nicht den gesamten Chat.

Eintrag	Das notierst du knapp in eigenen Worten
6 · Eigener nächster Schritt	Was habe ich danach selbst verändert oder ergänzt?
7 · Prüfung	Mit welchem Testfall oder Gegenbeispiel habe ich meinen Gedanken geprüft?
8 · Erkenntnis	Welcher Irrweg wurde korrigiert? Was ist noch offen?

HINWEISSTUFE KENNZEICHNEN

0 = ohne KI-Hilfe · 1 = Diagnosefrage · 2 = Strukturhinweis · 3 = kleines Teilbeispiel mit anderen Daten. Notiere die niedrigste Stufe, mit der du selbst weiterarbeiten konntest.

GEMEINSAME AUSWERTUNG

Ordne die Hürde am Ende möglichst einer Kategorie zu: Begriff/Datentyp · Ablauf/Reihenfolge · Bedingung · Schleife/Abbruch · Index/Datenstruktur · Syntax/Umgebung · Testfall · Datenmodell. Im Unterricht können anonymisierte Beispiele gebündelt werden: Welche Hürden traten häufig auf? Welche Denkfrage half? Welche Erklärung führte in die Irre? Daraus entstehen bessere Hilfekarten und Denk-Prompts für die nächste Lerngruppe.

DATENSCHUTZ UND FACHLICHE VERANTWORTUNG

Dokumentiere keine privaten Daten und keine vollständigen Chatverläufe. Für eine gemeinsame oder digitale Auswertung genügt ein Kürzel beziehungsweise eine anonyme ID. Der Lerncoach lernt aus den Einträgen nicht automatisch. Erst eine bewusst ausgewählte, anonymisierte Auswertung kann zur Verbesserung der Aufgaben, Hinweise und späteren Lernplattform verwendet werden. Die technische Speicherung wird in Kapitel 6 als optionales Persistenzmodell vorbereitet.

ABITURBEZUG · OPERATORENLISTE 2.2

Für Struktogramme gilt die „Abiturprüfung Baden-Württemberg: Operatorenliste für Struktogramme“, Version 2.2 vom 01.09.2024. Verwende deren Operatoren und Beschriftungen exakt. Bedingungen nutzen `==`, `!=`, `<`, `<=`, `>`, `>=` sowie `AND`, `OR` und `NOT`. Python- und PHP-Syntax gehört erst in den anschließenden Quellcode.

Pflichtumfang klar trennen

BG: Arrays implementieren; dynamische Strukturen beschreiben, unterscheiden und modellieren. Such- und Sortieralgorithmen werden als Funktionen geplant, im Struktogramm dargestellt, in der festgelegten Kurssprache Python oder PHP implementiert und durch getrennte Aufrufe getestet.

BKWI1: dieselbe Grundlage funktionsorientiert in Python und PHP; zusätzlich Listenoperationen, eine praxisnahe Warteschlange und einen ersten persistenten Datenzugriff implementieren, optional im GitHub-Codespace.

BEREITGESTELLTE GRUNDLAGE

Verwendet werden die Informationsdokumente zur verketteten Liste, zum Stapelspeicher, zur Warteschlange und zum Baum, der Lehrplan als PDF sowie die vorhandenen Volleyball-Struktogramme. Struktogramme werden im Unterricht und in Klassenarbeiten mit dem `hus` Struktogrammer aus der Digitalen Schultasche erstellt. Die selbst zu konfigurierende `draw.io`-Erweiterung der Autorin bleibt eine optionale Lernhilfe außerhalb von Klassenarbeiten.

Kapitel 1: Datenstrukturen im Überblick

↪ Inhalt

DEIN KAPITELSTART

Ziel: du Array, verkettete Liste, Stapel, Warteschlange und Baum anhand ihrer Ordnung, Zugriffsregel und typischen Verwendung unterscheidest.

Situation und jetzt tun: Ein Teammanager, ein Druckersystem und ein Dateibaum speichern jeweils mehrere Elemente – aber nicht nach derselben Regel. Ordne jedem Beispiel eine geeignete Struktur zu und begründe deine Entscheidung.

MÖGLICHER STARTPROMPT · KI ALS DENKCOACH

Nenne mir nacheinander kurze Anwendungssituationen. Lass mich jeweils Ordnung, erlaubten Zugriff, wichtige Operationen und eine geeignete Datenstruktur begründen. Gib die Zuordnung erst nach meinem Versuch bekannt.

BG · 2 Std./Woche

BPE 7.3: Strukturen beschreiben und modellieren; keine Implementierung dynamischer Strukturen im Pflichtteil

BKWI · 6 Std./Woche

Grundlage für BKWI1; anschließend Operationen vergleichen und eine Struktur praktisch umsetzen

Warum brauchst du das später?

Die Auswahl der Datenstruktur ist eine fachliche Entwurfsentscheidung. Sie bestimmt, welche Operationen natürlich, verständlich und effizient möglich sind. Die Business-Logik wird zuerst als Funktionsvertrag und BW-Struktogramm festgelegt. BG implementiert anschließend in der Kurssprache Python oder PHP; BKWI in beiden Sprachen.

1.1 Eine Leitfrage statt einer Merklste

Eine Datenstruktur organisiert mehrere zusammengehörige Daten. Entscheidend ist nicht nur, welche Werte gespeichert werden, sondern wie auf sie zugegriffen wird: über eine Position, über Verweise, nur am oberen Ende, in Ankunftsreihenfolge oder entlang einer Hierarchie.

Datenstruktur	Ordnungs- und Zugriffsprinzip	Typische Operationen	Praxisbeispiel
Array / einfache Liste	geordnet; Zugriff über Index	lesen, setzen, durchlaufen	Volleyball-Kader
verkettete Liste	Knoten mit Daten und Verweis	Knoten einfügen, löschen	veränderliche Folge
Stapel / Stack	LIFO: zuletzt hinein, zuerst hinaus	push, pop	Rückgängig-Funktion
Warteschlange / Queue	FIFO: zuerst hinein, zuerst hinaus	enqueue, dequeue	Druck- oder Ticketaufträge
Baum	hierarchische Knoten und Kanten	Kind/Teilbaum finden, durchlaufen	Dateisystem, Organigramm

BG-PFLICHTGRENZE

Im BG werden die dynamischen Strukturen beschrieben, unterschieden und modelliert. Programmiert werden im Kern das eindimensionale Array beziehungsweise die verwendete Python-Liste sowie später die geforderten Such- und Sortieralgorithmen. Eine Implementierung von verketteter Liste, Stack, Queue und Baum ist hier keine Voraussetzung.

1.2 Darstellungsformen lesen: Array und verkettete Liste

Bevor du eine Datenstruktur selbst modellierst, liest du zwei vollständige Darstellungen desselben Volleyball-Kaders. Entscheidend ist nicht das Aussehen allein, sondern welche Zugriffsregel die Darstellung sichtbar macht.

Array: Werte besitzen nummerierte Positionen

Index 0	Index 1	Index 2	Index 3	Index 4
Aylin	Ben	Cem	Daria	Mina

Direkter Zugriff: `kader[2]` liefert Cem. Die Position ist durch den Index sichtbar; beim Einfügen in der Mitte müssen nachfolgende Werte verschoben werden.

Einfach verkettete Liste: Jeder Knoten kennt seinen Nachfolger

START →	Aylin →	Ben →	Cem →	Daria →	NULL
-------------------	---------------------	----------------	----------------	---------------------	-------------

Schrittweiser Zugriff: Um Cem zu erreichen, beginnt man bei START und folgt den Verweisen über Aylin und Ben. NULL kennzeichnet, dass kein weiterer Knoten folgt. Beim Einfügen werden passende Verweise verändert.

Frage	Array	verkettete Liste
Wo liegt Cem?	Index 2	im dritten erreichbaren Knoten
Wie erfolgt der Zugriff?	direkt über den Index	ab START den Verweisen folgen

Frage	Array	verkettete Liste
Was ändert sich beim Einfügen?	Werte können verschoben werden	Nachfolger-Verweise werden angepasst
Wie wird das Ende erkannt?	letzter gültiger Index	Verweis auf NULL

FACHLICHE ABGRENZUNG

Im BG werden Array und verkettete Liste gelesen, beschrieben, unterschieden und modelliert. Programmiert werden im Kern das eindimensionale Array und später die vorgeschriebenen Sortier- und Suchalgorithmen. Eine Implementierung der verketteten Liste bleibt eine mögliche BKWI-Vertiefung.

DARSTELLUNGSARTEN NICHT VERWECHSELN

Die Strukturkarte beziehungsweise Speicheransicht zeigt Aufbau, Indizes, Knoten und Verweise einer Datenstruktur. Das Struktogramm zeigt dagegen den Ablauf eines Algorithmus und verwendet im BG die Operatorenliste Version 2.2. Für das Arraybild und die verkettete Liste werden daher keine Struktogramm-Operatoren erzwungen; für Sortier-, Such- und Verarbeitungsabläufe schon. Dafür wird verbindlich der Struktogrammer verwendet; draw.io ist lediglich eine optionale Vorbereitungshilfe und in Klassenarbeiten nicht zugelassen.

Vom fertigen Beispiel zur eigenen Darstellung

1. Markiere im Array Indizes und Werte; markiere in der verketteten Liste START, Datenfelder, Verweise und NULL.
2. Ergänze in einer teilweise leeren Darstellung die Spielerin Mina zwischen Ben und Cem.
3. Zeichne einen neuen Vierer-Kader in beiden Darstellungsformen.
4. Erkläre, wie Cem jeweils gefunden und wie eine neue Spielerin eingefügt wird.
5. Entscheide für eine Startaufstellung und für häufiges Einfügen, welche Struktur naheliegt, und begründe die Wahl.

Denkcoach-Prompt: den Lernweg selbst steuern

BEREITGESTELLTER STARTPROMPT

Du bist mein Denkcoach für Datenstrukturen. Hilf mir schrittweise, den Volleyball-Kader Aylin, Ben, Cem und Daria zuerst als eindimensionales Array und anschließend als einfach verkettete Liste darzustellen. Stelle immer nur eine Frage und warte auf meine Antwort. Fordere zuerst meinen eigenen Vorschlag ein. Frage beim Array nach Indizes und Werten, bei der verketteten Liste nach START, Knoten, Datenfeld, Nachfolger-Verweis und NULL. Gib Hinweise in drei Stufen: Denkfrage, Strukturhinweis, konkretes Teilbeispiel. Zeige die vollständige Lösung erst nach meinem eigenen Versuch. Lass mich am Ende erklären, wie Cem erreicht und Mina zwischen Ben und Cem eingefügt wird.

Stufe	Arbeit am Prompt	Ziel
1 · nutzen	bereitgestellten Prompt unverändert erproben	Dialogregeln erkennen
2 · ergänzen	Thema, Daten und gewünschte Hilfestufe einsetzen	Promptbestandteile bewusst wählen
3 · formulieren	eigenen Denkcoach-Prompt schreiben	Lernprozess selbst steuern
4 · prüfen	Antwortverhalten mit Qualitätskriterien bewerten	Lernhilfe von Lösungsausgabe unterscheiden

ARBEITSAUFTRAG

Formuliere nach dem ersten Durchlauf einen eigenen Denkcoach-Prompt für eine neue Datenstruktur. Er muss Rolle, Lernziel, Vorwissen, schrittweises Vorgehen, gestufte Hinweise, Fehlerkorrektur, gewünschte Darstellung und abschließende Selbstkontrolle enthalten. Tausche den Prompt mit einer Partnerin oder einem Partner und prüft: Musste die lernende Person selbst denken oder lieferte die KI die Lösung zu früh?

SELBSTKONTROLLE

Ich kann Array und verkettete Liste grafisch lesen und selbst darstellen; Index, START, Knoten, Verweis und NULL korrekt verwenden; Zugriffs- und Einfügewege erklären; einen Denkcoach-Prompt so formulieren, dass er eigene Lösungsversuche, gestufte Hilfen und Selbstkontrolle einfordert.

1.3 Statisch, dynamisch und abstrakt

Begriff	Arbeitsdefinition	Prüfrage
statisch	Größe oder Speicheraufteilung ist im Modell festgelegt	Muss die Anzahl vorher bekannt sein?
dynamisch	Struktur kann während der Laufzeit wachsen oder schrumpfen	Werden Knoten/Elemente hinzugefügt oder entfernt?
abstrakter Datentyp	Verhalten wird durch erlaubte Operationen beschrieben	Welche Zugriffe sind fachlich erlaubt?

1.5 Strukturen modellieren

1. Array: Kästchen mit Indizes und gespeicherten Werten zeichnen.

2. Verkettete Liste: Anker, Knoten, Datenfeld, Verweis und NULL-Ende kennzeichnen.
3. Stapel: oberstes Element und LIFO-Richtung markieren.
4. Warteschlange: Vorder- und Hinterende sowie FIFO-Richtung markieren.
5. Baum: Wurzel, Eltern, Kinder, Blätter und Kanten benennen.

KERNAUFTRAG · IM UNTERRICHT VERBINDLICH · RICHTZEIT 45-60 MINUTEN

Erstelle eine Vergleichskarte zu den fünf Datenstrukturen. Ordne danach die Situationen Volleyball-Kader, Browser-Zurück, Druckaufträge, Dateiverzeichnis und häufiges Einfügen in eine Folge zu. Begründe jede Wahl mit Zugriffsregel und benötigten Operationen; eine Programmiersprache ist für diesen Auftrag nicht erforderlich.

ERWARTETES LERNPRODUKT

DS01_datenstrukturen_ueberblick als SVG/PNG oder eine gleichwertige handschriftliche Strukturkarte sowie eine begründete Zuordnungstabelle.

Gestufte Tutorhilfe

Stufe	Hinweis
1 · Denkfrage	Welche Reihenfolge oder Hierarchie verlangt die Situation?
2 · Struktur	Frage anschließend, an welcher Stelle gelesen, eingefügt oder entfernt werden muss.
3 · Konkret	Nutze Index → Array, LIFO → Stack, FIFO → Queue, Verweise → verkettete Liste, Hierarchie → Baum nur als Entscheidungshilfe und begründe trotzdem fachlich.

Prüfnachweis · ADS-K01-A01

- ☐ Alle fünf Strukturen sind mit ihrem Ordnungsprinzip dargestellt.
- ☐ Zentrale Bestandteile und Operationen sind korrekt benannt.
- ☐ Jede Anwendung ist nachvollziehbar zugeordnet.
- ☐ Mindestens eine ungeeignete Alternative wird mit Begründung ausgeschlossen.
- ☐ Die Darstellung ist sprachunabhängig und kann später in einer Lernplattform erzeugt werden.

Vertiefung · optional für schnelle Schüler

Vergleiche zwei Lösungen für dieselbe Situation, zum Beispiel Volleyball-Kader als Array und als verkettete Liste. Beurteile Zugriff über Position, Einfügen in der Mitte, Speicheraufwand und Verständlichkeit.

Transfer zum Learning Agent

Mein Lernagent

Funktionsbausteine bleiben einzeln aufrufbar und testbar

Aufgabe bearbeiten

Mein Lösungsversuch

Diagnose starten

Hinweis anfordern

Lernweg

- 1 Eingabe prüfen
- 2 Hürde erkennen
- 3 Hinweis wählen
- 4 Ergebnis testen

`prüfe_eingabe() → diagnostiziere() → wähle_hinweis() → prüfe_antwort()`

Abbildung 14: Lernagent mit getrennten Diagnose- und Hinweisfunktionen.

Entwickle `diagnose_sortierfehler` oder `diagnose_suchschritt` als Funktion. Sie erhält einen protokollierten Zustand und gibt genau eine Diagnosefrage zurück, keine Komplettlösung.

AGENTENBAUSTEIN · KURZSICHERUNG 5-10 MINUTEN

Entwickle für den Learning Agent eine Entscheidungshilfe. Sie fragt nacheinander nach Reihenfolge, direktem Zugriff, Einfüge-/Löschhäufigkeit und Hierarchie und fordert anschließend eine begründete Strukturwahl – ohne sofort eine Lösung vorzugeben.

HALTE FEST

Benötigte Daten · Zugriffsregel · Operationen · passende Struktur · ausgeschlossene Alternative · Begründung

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich bearbeite die Auswahl einer geeigneten Datenstruktur. Mein Anwendungsfall ist: [einsetzen]. Mein bisheriger Vorschlag ist: [einsetzen]. Frage mich zuerst nach benötigten Zugriffen, Einfüge- und Entfernungsoperationen sowie der Reihenfolge. Gib danach immer nur eine Hilfestufe und warte auf meine Antwort. Stufe 1: eine Denkfrage. Stufe 2: ein Hinweis auf das passende Ordnungsprinzip. Stufe 3: ein kleines Gegenbeispiel mit anderen Daten. Nenne die geeignete Datenstruktur nicht, bevor ich mich selbst entschieden und meine Wahl begründet habe. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

Mini-Check

Warum lässt sich eine Datenstruktur nicht allein danach auswählen, dass sie ‚mehrere Werte speichern‘ kann?

Verbindlicher Funktionspfad für Algorithmen

Alle Operationen und Algorithmen werden als aufrufbare Funktionen beziehungsweise Methoden entwickelt. Ein Vertrag benennt Parameter, Rückgabe und mögliche Zustandsänderungen. Das Struktogramm bildet die Funktion oder Methode ab; der Testaufruf steht außerhalb.

BG: Die Lehrkraft legt Python oder PHP als Kurssprache fest. Bewertet werden das sprachneutrale Struktogramm, die Implementierung in dieser einen Sprache und aussagekräftige Testaufrufe. BKWI: Dasselbe Struktogramm wird in Python und PHP umgesetzt; die Lernenden vergleichen beide Aufrufe und Ergebnisse.

Benutzeroberflächen zeigen den Anwendungskontext. Sie rufen Funktionen auf, enthalten aber nicht selbst die Sortier-, Such- oder Datenstruktur-Logik.

Der Lernagent verlangt zuerst einen eigenen Lösungsversuch und unterstützt anschließend mit Diagnosefrage und gestuften Hinweis. Der eigene Test und das Lernwegprotokoll mit Ausgangsidee, Änderung, Ergebnis und offener Frage schließen den Lernweg ab.

Schritt	Nachweis
1 Funktionsvertrag	Name, Parameter, Rückgabe, Veränderung der übergebenen Daten
2 Struktogramm	Funktions- oder Methodenrahmen mit BW-Operatoren 2.2
3 Implementierung	BG eine Kurssprache; BKWI Python und PHP
4 Testaufruf	Normalfall, Randfall, Fehler- oder Nicht-gefunden-Fall
5 Lernagent	Diagnosefrage, gestufter Hinweis, eigener Test und Lernwegprotokoll

Kapitel 2: Operationen vergleichen und auswählen ↩ Inhalt

DEIN KAPITELSTART

Ziel: du Einfügen, Entfernen, Lesen, Suchen und Tauschen an einer geordneten Sammlung planst und daraus eine begründete Strukturwahl ableitest.

Situation und jetzt tun: Im Volleyball-Teammanager soll eine Spielerin gesucht, an einer Position eingefügt oder entfernt werden; außerdem sollen zwei Positionen getauscht werden. Zerlege jede Anforderung in Daten, Positionen und Operationen.

MÖGLICHER STARTPROMPT · KI ALS DENKCOACH

Arbeite mit mir an einer Listenoperation. Frage zuerst nach Vorbedingung, betroffener Position, Veränderung der Länge, Ergebnis und Fehlerfall. Zeige erst danach einen sprachneutralen Ablauf und auf Wunsch Code.

BG · 2 Std./Woche

BPE 7.1/7.3: Array-Operationen anwenden; dynamische Strukturen weiterhin beschreiben und modellieren

BKWI · 6 Std./Woche

BKWI1: dieselben Operationen in Python und PHP umsetzen und mit den vorhandenen Volleyball-Struktogrammen vergleichen

Warum brauchst du das später?

Operationen verbinden Datenstruktur und Algorithmus. Wer zuerst Vorbedingungen und Auswirkungen klärt, kann vorhandene Bibliotheksoperationen sinnvoll einsetzen oder den Ablauf selbst modellieren und testen.

2.1 Operationen als Vertrag

Operation	Vorbedingung	Wirkung	Typischer Fehlerfall
lesen	Index ist vorhanden	Wert bleibt unverändert	Index außerhalb der Sammlung
einfügen	Position ist zulässig	Länge wächst; Nachfolger verschieben sich	falsche Position
entfernen	Element/ Index existiert	Länge sinkt	Element nicht gefunden
suchen	Suchkriterium ist definiert	Index oder Nicht-gefunden-Ergebnis	mehrdeutige Namen
tauschen	beide Indizes existieren	Länge bleibt gleich	gleiche/ ungültige Position

2.2 Direkter Sprachvergleich am Kader

Volleyball-Teammanager

Oberfläche für Liste, Suche und Positionswechsel

Startaufstellung	Aktionen
1 Ada	<button>Spieler suchen</button>
2 Bo	<button>An Position einfügen</button>
3 Cem	<button>Positionen tauschen</button>
4 Dana	<button>Spieler entfernen</button>
5 Eli	
6 Flo	

Abbildung 1: Teammanager als gemeinsame Oberfläche mehrerer Listenoperationen.

Wähle zu jeder Aktion eine geeignete Operation und begründe, welche Vor- und Nachbedingungen gelten.

Python · BG bei Kurs Sprache Python · BKWI

```
def kader_bearbeiten(kader):
    ergebnis = kader.copy()
    ergebnis.insert(1, "Daria")
    entfernt = ergebnis.pop(2)
    ergebnis[0], ergebnis[1] =
    ergebnis[1], ergebnis[0]
    return ergebnis, entfernt

print(kader_bearbeiten(["Aylin",
"Ben", "Cem"]))
```

PHP · BG bei Kurs Sprache PHP · BKWI

```
<?php
function kaderBearbeiten($kader) {
    $ergebnis = $kader;
    array_splice($ergebnis, 1, 0,
    ["Daria"]);
    $entfernt =
    array_splice($ergebnis, 2, 1)[0];
    [$ergebnis[0], $ergebnis[1]] =
    [$ergebnis[1], $ergebnis[0]];
    return [$ergebnis, $entfernt];
}
print_r(kaderBearbeiten(["Aylin",
"Ben", "Cem"]));
```

GLEICHE FACHLOGIK, ANDERE BIBLIOTHEK

Einfügen, Entfernen und Tauschen verändern in beiden Fassungen dieselbe geordnete Sammlung. Python und PHP stellen dafür unterschiedliche Operationen bereit. Das gemeinsame Struktogramm beschreibt die Wirkung, nicht den Namen einer Bibliotheksfunktion.

2.3 Geeignet auswählen

1. Daten und fachliche Reihenfolge benennen.
2. Häufige Operationen und erlaubte Zugriffe festlegen.
3. Randfälle bestimmen: leer, voll, ungültige Position, nicht gefunden.
4. Mindestens zwei Strukturen vergleichen.
5. Auswahl mit einem fachlichen Vorteil und einem Nachteil begründen.

Situation	naheliegende Wahl	Begründung
Startaufstellung mit festen Positionen	Array / Liste	direkter Zugriff über Positionsindex
letzte Aktion rückgängig machen	Stack	zuletzt ausgeführte Aktion zuerst entnehmen
Anfragen fair bearbeiten	Queue	Ankunftsreihenfolge bleibt erhalten
Ordner und Unterordner	Baum	hierarchische Beziehungen
häufiges Umhängen von Nachfolgern	verkettete Liste	Verweise statt zusammenhängender Positionen

KERNAUFTRAG · IM UNTERRICHT VERBINDLICH · RICHTZEIT 45-60 MINUTEN

Setze den Volleyball-Teammanager als zusammenhängende Aufgabenfamilie fort: Kader anzeigen, Spieler nach Name suchen, an einer gültigen Position einfügen, entfernen und zwei Positionen tauschen. Nutze die bereitgestellten Struktogramme als Vergleichsgrundlage. BG: mindestens zwei Operationen in Python und alle Abläufe erklären. BKWI: alle Operationen in Python und PHP umsetzen und mit Normal-, Rand- und Fehlerfällen testen.

ERWARTETES LERNPRODUKT

DS02_volleyball_teammanager.py; im BKWI zusätzlich .php; überarbeitete Struktogramme-Originaldateien mit SVG/PNG-Export und eine Operationen-Testtabelle.

Gestufte Tutorhilfe

Stufe	Hinweis
1 · Denkfrage	Verändert die Operation die Länge, die Reihenfolge oder nur einen gelesenen Wert?
2 · Struktur	Prüfe Indizes beziehungsweise Suchergebnis, bevor du einfügst, entfernst oder tauschst.
3 · Konkret	Trenne jede Operation in eine eigene Funktion und gib Erfolg oder Fehler als Rückgabewert zurück.

Prüfnachweis · ADS-K02-A01

- ☐ Jede Operation besitzt klar benannte Vorbedingungen.
- ☐ Ungültige Positionen und Nicht-gefunden-Fälle werden behandelt.
- ☐ Die Reihenfolge nach Einfügen, Entfernen und Tauschen ist korrekt.
- ☐ Die bereitgestellten Volleyball-Struktogramme sind sichtbar eingebunden oder begründet überarbeitet.

- ☐ BG- und BKWI-Pflichtumfang sind klar getrennt.
- ☐ Im BKWI bilden Python und PHP dieselbe Funktion mit identischem Vertrag und identischen Testfällen ab.

Vertiefung · optional für schnelle Schüler

Entkopple die Benutzerführung von den Listenfunktionen. Jede Listenoperation wird als Funktion mit Parametern und Rückgabe entwickelt; ein getrenntes Hauptprogramm übernimmt Aufruf, Ein- und Ausgabe. Bereite damit den späteren Übergang zu Klassen und einer Weboberfläche vor.

Transfer zum Learning Agent

AGENTENBAUSTEIN · KURZSICHERUNG 5-10 MINUTEN

Übertrage die Operationen auf Lernstände: Versuch ergänzen, letzten Eintrag lesen, einen Fehlercode suchen und zwei geplante Aufgaben umordnen. Entscheide für jede Teilaufgabe, ob Liste, Stack oder Queue die passendere Sicht ist.

HALTE FEST

Vorbedingung · Operation · Rückgabewert · Fehlerfall · Test · begründete Strukturwahl

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich arbeite an einer Listenoperation im Volleyball-Teammanager. Operation: [lesen/einfügen/entfernen/suchen/tauschen]. Mein aktueller Stand: [einsetzen]. Erwartetes Ergebnis: [einsetzen]. Prüfe zuerst mit einer Frage, ob ich Vorbedingung, betroffene Position und Nachbedingung verstanden habe. Hilf danach schrittweise, aber schreibe weder die fertige Funktion noch vollständigen Code. Lass mich zum Schluss einen Normalfall und einen Fehlerfall selbst formulieren. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

Mini-Check

Welche Information muss vor einer Listenoperation geprüft werden, damit kein ungültiger Zugriff entsteht?

Kapitel 3: Warteschlange praktisch umsetzen · BKWI [↔ Inhalt](#)

DEIN KAPITELSTART

Ziel: du das FIFO-Prinzip modellierst und im BKWI mit einer geeigneten Bibliotheksstruktur in Python und PHP implementierst.

Situation und jetzt tun: Ein Ticketsystem nimmt Anfragen in ihrer Ankunftsreihenfolge auf. Neue Tickets kommen hinten hinzu; bearbeitet wird immer das älteste offene Ticket. Erkläre, warum weder Positionswahl noch LIFO dazu passen.

MÖGLICHER STARTPROMPT · KI ALS DENKCOACH

Lass mich eine Warteschlange mit Vorder- und Hinterende zeichnen. Gib mir ENQUEUE- und DEQUEUE-Schritte und frage nach dem Zustand nach jeder Operation. Zeige Bibliothekscode erst, wenn meine FIFO-Folge stimmt.

BG · 2 Std./Woche

Transfer: FIFO beschreiben, Zustandstabellen und Modelle lesen; keine Pflichtimplementierung

BKWI · 6 Std./Woche

BKWI1: verbindliche Zusatzimplementierung in Python und PHP, optional im GitHub-Codespace

Warum brauchst du das später?

Warteschlangen sind in Auftragsbearbeitung, Kassen, Drucksystemen und technischen Puffern unmittelbar praxisrelevant. Deshalb wird diese dynamische Struktur im BKWI zusätzlich implementiert; im BG bleibt die Beschreibung und Modellierung ausreichend.

3.1 FIFO und erlaubte Zugriffe

ENQUEUE fügt ein Element am Hinterende hinzu. DEQUEUE entfernt das Element am Vorderende. Die fachliche Regel verhindert, dass

ein später eingetroffener Auftrag ohne besondere Prioritätsregel vorgezogen wird.

Schritt	Operation	Warteschlange vorn → hinten	Ergebnis
1	ENQUEUE(A101)	A101	-
2	ENQUEUE(A102)	A101, A102	-
3	ENQUEUE(A103)	A101, A102, A103	-
4	DEQUEUE()	A102, A103	A101

3.2 Python und PHP mit geeigneten Bibliotheksstrukturen

Python · BG bei Kurssprache Python · BKWI

```
from collections import deque

def naechstes_ticket(tickets):
    queue = deque(tickets)
    if not queue:
        return None
    return queue.popleft()

print(naechstes_ticket(["A101",
"A102"]))
```

PHP · BG bei Kurssprache PHP · BKWI

```
<?php
function naechstesTicket($tickets) {
    $queue = new SplQueue(); foreach
($tickets as $ticket) { $queue-
>enqueue($ticket); }
    if ($queue->isEmpty()) { return
null; }
    return $queue->dequeue();
}
echo naechstesTicket(["A101",
"A102"]);
```

WARUM DIESE UMSETZUNG?

Python deque unterstützt effizientes Anhängen und Entfernen an beiden Enden; für FIFO werden append und popleft genutzt. PHP stellt mit SplQueue direkt die Operationen enqueue und dequeue bereit. In beiden Programmen muss vor DEQUEUE geprüft werden, ob die Warteschlange leer ist.

3.3 Von der Automatenaufgabe zum Auftragssystem

Auftragswarteschlange

FIFO: zuerst eingegangen – zuerst bearbeitet



Nächster Auftrag

A-104 · Passwort zurücksetzen

Bearbeiten

Abbildung 2: Auftragsansicht einer FIFO-Warteschlange.

Produziere nach jeder vorgegebenen enqueue- oder dequeue-Operation den sichtbaren Zustand der Oberfläche.

Getränke- oder Milchausgaben können zunächst unmittelbar verarbeitet werden. Sobald mehrere Bestellungen eintreffen, trennt eine Warteschlange die Annahme von der Bearbeitung: Bestellung aufnehmen, hinten einreihen, älteste Bestellung prüfen, Bestand verändern und Ergebnis ausgeben.

KERNAUFTRAG · IM UNTERRICHT VERBINDLICH · RICHTZEIT 45-60 MINUTEN

Implementiere im BKWI eine Warteschlange für Getränke- oder Supportaufträge. Das Programm bietet die Aktionen Auftrag aufnehmen, nächsten Auftrag bearbeiten, Warteschlange anzeigen und beenden. Leere Entnahmeversuche werden abgefangen. Erstelle zuerst Zustandsdiagramm beziehungsweise Strukturkarte und Struktogramm, danach Python- und PHP-Fassung mit identischen Testfolgen. BG: bearbeite nur Modell, Zustandsfolge und Begründung der Strukturwahl.

ERWARTETES LERNPRODUKT

DS03_warteschlange als Strukturkarte sowie Struktogrammer-Originaldatei mit SVG/PNG-Export, Zustandstabelle und im BKWI DS03_warteschlange.py sowie DS03_warteschlange.php.

Gestufte Tutorhilfe

Stufe	Hinweis
1 · Denkfrage	Welches Element muss als Nächstes bearbeitet werden?
2 · Struktur	Trenne Aufnahme mit ENQUEUE und Bearbeitung mit DEQUEUE; prüfe vor der Entnahme den Leerfall.
3 · Konkret	Python: deque, append, popleft. PHP: SplQueue, enqueue, dequeue und isEmpty.

Prüfnachweis · ADS-K03-A01

- ☐ Vorder- und Hinterende sind im Modell eindeutig markiert.
- ☐ Die Testfolge hält FIFO ohne Ausnahme ein.
- ☐ Ein leerer Entnahmeversuch führt nicht zu einem Programmfehler.
- ☐ Aufnahme, Bearbeitung, Anzeige und Abbruch sind getrennte Aktionen.
- ☐ Im BKWI liefern Python und PHP für dieselbe Befehlsfolge dieselbe Bearbeitungsreihenfolge.

- ☐ Die Wahl der Queue wird gegenüber Stack und einfacher Positionsliste begründet.

Vertiefung · optional für schnelle Schüler

Ergänze Wartezeit oder Eingangszeitpunkt und berechne nach jeder Bearbeitung die verbleibende Anzahl. Eine Prioritätswarteschlange ist ein späterer Ausblick und darf die FIFO-Regel nur aufgrund einer ausdrücklich formulierten Fachregel verändern.

Transfer zum Learning Agent

AGENTENBAUSTEIN · KURZSICHERUNG 5-10 MINUTEN

Modelliere offene Lernanfragen als Warteschlange. Neue Fragen werden aufgenommen, die älteste offene Frage wird zuerst bearbeitet, und der Leerfall erzeugt eine ruhige Statusmeldung. Halte zusätzlich fest, wann eine Prioritätsregel fachlich gerechtfertigt wäre.

HALTE FEST

ENQUEUE-Ereignis · DEQUEUE-Regel · Leerfall · sichtbare Rückmeldung · Testfolge · Fairnessbegründung

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich modelliere die Warteschlange des Ausgabeautomaten. Mein letzter sicherer Zustand ist: [Inhalt der Queue]. Die nächste Aktion ist: [ENQUEUE/DEQUEUE]. Frage mich zuerst, an welchem Ende gearbeitet wird und welches FIFO-Prinzip gilt. Gib nur dann einen weiteren Hinweis, wenn ich darum bitte. Nutze für ein Teilbeispiel andere Elemente als in meiner Aufgabe. Gib keine fertige Zustandsfolge und keinen vollständigen Programmcode aus. Lass mich am Ende den Rückgabewert und den neuen Zustand erklären. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

Mini-Check

Woran erkennst du in einer Zustandsfolge eindeutig, dass eine Warteschlange und kein Stapel verwendet wurde?

Kapitel 4: Sortieren - Selection Sort und Bubble Sort

↪ Inhalt

KAPITELZIEL

Du erläuterst Selection Sort und Bubble Sort an einem eindimensionalen Array, führst beide Verfahren als Schreibtischtest aus und implementierst sie in Python beziehungsweise im BKWI zusätzlich in PHP. Du vergleichst die Verfahren anhand der benötigten Vergleiche und Vertauschungen.

Ausgangssituation: Die Mannschaftsliste ordnen

Der Volleyball-Teammanager speichert die Namen zunächst in einem einfachen Array: Mina, Ben, Aylin, Cem, Daria. Für eine alphabetische Meldeliste soll der Inhalt ohne eingebaute Sortierfunktion geordnet werden. Die Elemente bleiben im selben Array; nur ihre Positionen ändern sich.

ARBEITSAUFTRAG

Lege fünf Namenskarten in der angegebenen Reihenfolge aus. Sortiere sie zunächst von Hand. Protokolliere jeden Vergleich und jeden Tausch. Formuliere danach eine Regel, die nicht nur für diese fünf Namen, sondern für beliebig viele Arrayelemente gilt.

LEHRPLANBEZUG UND UMFANG

BPE 7.2: Sortieralgorithmen an der Datenstruktur Array erläutern, anwenden und implementieren. Reihenfolge: Selection Sort, anschließend Bubble Sort. BG arbeitet verbindlich in Python; BKWI1 setzt dieselbe Fachlogik zusätzlich in PHP um.

4.1 Vor dem Sortieren: Vertrag und Tausch

Sortierlabor

Algorithmus als Funktion aufrufen und Zustände prüfen

The screenshot shows a web application titled 'Sortierlabor'. It has a light gray background. At the top, there are two labels: 'Eingabeliste' and 'Verfahren'. Below 'Eingabeliste' is a text input field containing the string '5, 2, 8, 1'. Below 'Verfahren' is a dropdown menu showing 'Selection Sort' with a downward arrow. In the center, there is a dark green button labeled 'Sortierfunktion testen'. To the right of the button is a light green box containing the text 'Ausgabe: 1, 2, 5, 8' and 'Vergleiche: 6'. At the bottom of the interface, there is a red text link: 'Testfälle: normal · bereits sortiert · ein Element · leer'.

Abbildung 8: Sortierlabor mit auswählbarer Algorithmusfunktion.

Leite aus der Oberfläche den Funktionsvertrag ab. Der Algorithmus darf keine Eingaben einlesen, sondern erhält die Liste als Parameter und gibt eine sortierte Liste zurück.

Eingabe ist ein eindimensionales Array vergleichbarer Werte. Ausgabe ist dasselbe Array in aufsteigender Reihenfolge. Für den Unterricht werden bewusst weder die Python-Funktion `sorted()` noch die Sortiermethoden beziehungsweise -funktionen `sort()` verwendet, weil der Ablauf des Algorithmus sichtbar bleiben soll.

Python	PHP
<pre>def tauschen(kader, i, j): ergebnis = kader.copy() ergebnis[i], ergebnis[j] = ergebnis[j], ergebnis[i] return ergebnis print(tauschen(["A", "B", "C"], 0, 2))</pre>	<pre><?php function tauschen(\$kader, \$i, \$j) { \$ergebnis = \$kader; [\$ergebnis[\$i], \$ergebnis[\$j]] = [\$ergebnis[\$j], \$ergebnis[\$i]]; return \$ergebnis; } print_r(tauschen(["A", "B", "C"], 0, 2));</pre>

SELBSTKONTROLLE

Nach einem Tausch enthält das Array dieselben Werte wie vorher. Nur die Positionen i und j sind vertauscht; die Länge bleibt unverändert.

4.2 Selection Sort: jeweils das kleinste Element auswählen

Funktions-Struktogramm

BW-konforme Funktionsdarstellung und separater Testaufruf

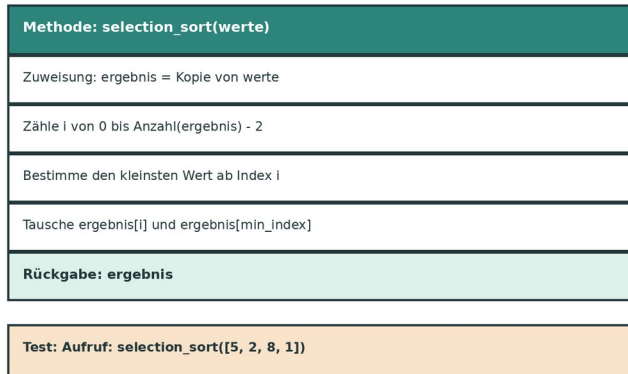


Abbildung 9: Funktions-Struktogramm von Selection Sort mit getrenntem Testaufruf.

Prüfe, ob jeder Operator innerhalb der Funktion liegt und ob der Aufruf außerhalb steht. Ergänze einen Randfall mit leerer Liste.

Selection Sort verfolgen

Nach jedem äußeren Durchlauf wächst der sortierte Bereich

Schritt 0	5	2	8	1	Minimum suchen
Schritt 1	1	2	8	5	Minimum suchen
Schritt 2	1	2	8	5	Minimum suchen
Schritt 3	1	2	5	8	

Abbildung 3: Zustandsfolge von Selection Sort.

Prüfe die Zustandsfolge. Markiere für jeden äußeren Durchlauf Suchbereich, gefundenes Minimum und Tauschpartner.

Selection Sort teilt das Array gedanklich in einen sortierten linken und einen noch unsortierten rechten Bereich. Für jede Startposition wird im restlichen Bereich das kleinste Element gesucht und anschließend an die Startposition getauscht.

Durchlauf	Start / Minimum	Array nach dem Tausch	Sortierter Bereich
1	0 / Aylin an Index 2	Aylin, Ben, Mina, Cem, Daria	Index 0
2	1 / Ben an Index 1	Aylin, Ben, Mina, Cem, Daria	Index 0-1
3	2 / Cem an Index 3	Aylin, Ben, Cem, Mina, Daria	Index 0-2

Durchlauf	Start / Minimum	Array nach dem Tausch	Sortierter Bereich
4	3 / Daria an Index 4	Aylin, Ben, Cem, Daria, Mina	Index 0-3

Python	PHP
<pre>def selection_sort(kader): n = len(kader) for start in range(n - 1): min_index = start for i in range(start + 1, n): if kader[i] < kader[min_index]: min_index = i if min_index != start: kader[start], kader[min_index] = (kader[min_index], kader[start]) kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"] selection_sort(kader) print(kader)</pre>	<pre><?php function selectionSort(array &\$kader): void { \$n = count(\$kader); for (\$start = 0; \$start < \$n - 1; \$start++) { \$minIndex = \$start; for (\$i = \$start + 1; \$i < \$n; \$i++) { if (\$kader[\$i] < \$kader[\$minIndex]) { \$minIndex = \$i; } } if (\$minIndex != \$start) { \$temp = \$kader[\$start]; \$kader[\$start] = \$kader[\$minIndex]; \$kader[\$minIndex] = \$temp; } } \$kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"]; selectionSort(\$kader); print_r(\$kader);</pre>

HILFESTELLUNG

Die äußere Schleife legt die nächste Zielposition fest. Die innere Schleife sucht nur den Index des kleinsten Elements. Getauscht wird erst nach Abschluss der inneren Schleife. Kommentiere insbesondere die Aufgaben von start, i und min_index beziehungsweise minIndex.

ARBEITSAUFTRAG

Übertrage die vier Durchläufe in ein Struktogramm nach Operatorenliste Version 2.2. Verwende die zulässige Zählschleife, den Wenn-Operator, Zuweisungen und Arrayzugriffe in der offiziellen Schreibweise. Markiere äußere Schleife, innere Schleife, Minimumvergleich und bedingten Tausch. Teste anschließend ein bereits sortiertes, ein umgekehrt sortiertes Array und ein Array mit zwei gleichen Namen.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich versuche Selection Sort für das Array [Werte einsetzen] nachzuvollziehen. Mein letzter sicherer Stand ist Durchlauf [Nummer] mit dem Array [Zustand]. Frage mich zuerst nach Startposition, unsortiertem Bereich und bisherigem Minimum. Hilf bei der Modellierung ausschließlich mit Operatoren aus Version 2.2, aber gib nur den nächsten benötigten Operator an. Berechne nicht den nächsten vollständigen Arrayzustand, zeichne kein fertiges Struktogramm und liefere keinen fertigen Code. Lass mich den nächsten Vergleich und einen möglichen Tausch selbst begründen. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

4.3 Bubble Sort: große Elemente nach rechts aufsteigen lassen

Bubble Sort vergleicht benachbarte Elemente. Stehen sie in falscher Reihenfolge, werden sie sofort getauscht. Nach einem vollständigen Durchlauf steht das größte noch unsortierte Element rechts. Wenn in einem Durchlauf kein Tausch stattfindet, ist das Array bereits sortiert.

Phase	Wichtige Vergleiche	Array am Phasenende	Erkenntnis
1	Mina mit Ben, Aylin, Cem, Daria	Ben, Aylin, Cem, Daria, Mina	Mina steht endgültig rechts.

Phase	Wichtige Vergleiche	Array am Phasenende	Erkenntnis
2	Ben mit Aylin (Tausch), danach Ben mit Cem und Cem mit Daria	Aylin, Ben, Cem, Daria, Mina	Die Liste ist geordnet; wegen des Tauschs folgt noch ein Prüfdurchlauf.
3	Aylin mit Ben und Ben mit Cem	Aylin, Ben, Cem, Daria, Mina	Kein Tausch: vorzeitiger Abbruch.

Python	PHP
<pre>def bubble_sort(kader): n = len(kader) for ende in range(n - 1, 0, -1): getauscht = False for i in range(ende): if kader[i] > kader[i + 1]: kader[i], kader[i + 1] = (kader[i + 1], kader[i]) getauscht = True if not getauscht: break kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"] bubble_sort(kader) print(kader)</pre>	<pre><?php function bubbleSort(array &\$kader): void { \$n = count(\$kader); for (\$ende = \$n - 1; \$ende > 0; \$ende--) { \$getauscht = false; for (\$i = 0; \$i < \$ende; \$i++) { if (\$kader[\$i] > \$kader[\$i + 1]) { \$temp = \$kader[\$i]; \$kader[\$i] = \$kader[\$i + 1]; \$kader[\$i + 1] = \$temp; \$getauscht = true; } } if (!\$getauscht) { break; } } \$kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"]; bubbleSort(\$kader); print_r(\$kader); }</pre>

HILFESTELLUNG

ende grenzt den noch unsortierten Bereich ab. i und $i + 1$ bezeichnen immer ein Nachbarpaar. Die Variable getauscht ermöglicht den vorzeitigen Abbruch, verändert aber nicht die fachliche Sortierregel.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich untersuche Bubble Sort und hänge bei [Durchlauf/Schleifengrenze/Abbruchbedingung]. Mein Arrayzustand ist [einsetzen]. Stelle zuerst genau eine Frage dazu, welches Nachbarpaar als Nächstes verglichen wird und welcher Bereich bereits sicher sortiert ist. Verwende im Struktogramm nur Zählschleife, Wenn-Operator, Zuweisung und Arrayzugriff nach Version 2.2; gib jeweils nur den nächsten Operator an. Schreibe weder die vollständige Zustandsfolge noch ein fertiges Struktogramm oder Programm. Prüfe am Ende mit mir den vorzeitigen Abbruch. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

4.4 Verfahren vergleichen und begründet auswählen

Merkmal	Selection Sort	Bubble Sort
Grundidee	Minimum im Rest suchen, danach einmal tauschen	Nachbarn vergleichen und sofort tauschen
Sicherer Bereich	links wächst der sortierte Bereich	rechts wächst der sortierte Bereich
Vergleiche	auch bei vorsortierten Daten viele Vergleiche	mit Abbruch bei vorsortierten Daten früh fertig

Merkmal	Selection Sort	Bubble Sort
Vertauschungen	höchstens ein Tausch je äußerem Durchlauf	je Durchlauf mehrere Tausche möglich
Typischer Aufwand	quadratisch bei wachsender Arraylänge	quadratisch; optimierter Best Case linear

BEWERTUNG

Für große reale Datenbestände werden leistungsfähigere Bibliotheksalgorithmen verwendet. Selection Sort und Bubble Sort sind hier Lernmodelle: An ihnen werden Schleifen, Vergleiche, Invarianten, Tauschoperationen und Laufzeitbeobachtung sichtbar.

Kapitelaufgabe: Kaderlisten sortieren · ADS-K04-A01

ARBEITSAUFTRAG

Implementiere beide Verfahren ohne eingebaute Sortierfunktion. Das Programm liest fünf Spielernamen über die Shell ein, zeigt nach jedem äußeren Durchlauf den Arrayzustand und gibt am Ende die alphabetische Liste aus. Zähle Vergleiche und Tausche. Prüfe dieselben Testdaten mit beiden Algorithmen.

Python	PHP
<pre>def kader_einlesen(anzahl): kader = [] for nummer in range(1, anzahl + 1): kader.append(input(f"Name {nummer}: ").strip()) return kader print(kader_einlesen(5))</pre>	<pre><?php function kaderEinlesen(\$anzahl) { \$kader = []; for (\$nummer = 1; \$nummer <= \$anzahl; \$nummer++) { \$kader[] = trim(readline("Name \$nummer: ")); } return \$kader; } print_r(kaderEinlesen(5));</pre>

Abgabeprodukte: DS04_selection_sort und DS04_bubble_sort als Struktogrammer-Originaldateien mit SVG/PNG-Export, DS04_sortieren.py; im BKWI1 zusätzlich DS04_sortieren.php sowie Schreibtischtest und Vergleichstabelle.

HILFESTELLUNG

Basis: Vervollständige einen vorgegebenen Algorithmus und nutze die fünf Beispielnamen. Standard: Implementiere beide Verfahren mit Ausgabe der Zwischenstände. Plus: Zähle Vergleiche und Tausche für verschiedene Eingabereihenfolgen und erkläre die Beobachtung.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich implementiere [Selection Sort/Bubble Sort] in [Python/PHP]. Mein Codeausschnitt oder Pseudocode lautet: [einsetzen]. Erwartet wird: [einsetzen]. Tatsächlich erhalte ich: [einsetzen]. Fordere mich zuerst auf, die Aufgabe jeder Schleife und die gültigen Indexgrenzen zu erklären. Hilf mir danach mit einer Diagnosefrage pro Schritt. Zeige höchstens eine zu korrigierende Zeile oder ein Mini-Beispiel mit anderen Daten, aber keine vollständige Lösung. Lass mich die Korrektur selbst schreiben und anschließend mit leerem, vorsortiertem und umgekehrt sortiertem Array prüfen. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

AUSBLICK LEARNING AGENT

Der Learning Agent lässt Lernende zuerst den nächsten Arrayzustand vorhersagen. Erst danach zeigt er den ausgeführten Vergleich, einen möglichen Tausch und die Begründung. So prüft er den Ablauf statt nur das Endergebnis.

SELBSTKONTROLLE

Ich kann den sortierten und unsortierten Bereich kennzeichnen;
Selection Sort und Bubble Sort schrittweise ausführen; beide
Algorithmen mit verschachtelten Schleifen implementieren; Variablen
und Abbruchbedingungen erklären; Verfahren mit Testfällen vergleichen.

Kapitel 5: Suchen - lineare und binäre Suche

Suche ↪ Inhalt

KAPITELZIEL

Du erläuterst lineare und binäre Suche an einem eindimensionalen Array, implementierst beide Verfahren und behandelst den Nicht-gefunden-Fall. Du begründest, warum die binäre Suche nur auf einem sortierten Array korrekt arbeitet.

Ausgangssituation: Eine Spielerin finden

Der Volleyball-Teammanager soll zu einem eingegebenen Namen die Position im Kader ausgeben. Eine unsortierte Trainingsliste kann direkt linear durchsucht werden. Für die binäre Suche muss zuerst eine alphabetisch sortierte Meldeliste vorliegen.

ARBEITSAUFTRAG

Suche Daria zuerst in der unsortierten Liste Mina, Ben, Aylin, Cem, Daria und danach in der sortierten Liste Aylin, Ben, Cem, Daria, Mina. Notiere in beiden Fällen die geprüften Positionen. Welche Voraussetzung nutzt das zweite Vorgehen?

LEHRPLANBEZUG UND UMFANG

BPE 7.2: Suchalgorithmen an der Datenstruktur Array erläutern, anwenden und implementieren. Reihenfolge: lineare Suche, anschließend binäre Suche. BG programmiert in Python; BKWI1 überträgt dieselbe Fachlogik zusätzlich nach PHP.

5.1 Ein eindeutiger Suchvertrag

Spielsuche

Lineare und binäre Suchfunktion mit demselben Vertrag

Spielername

Suchen

Gefunden an Index 3

Rückgabe der Funktion: 3

Gleicher Aufruf - unterschiedliche innere Strategie

Abbildung 10: Spielsuche als Oberfläche für zwei austauschbare Suchfunktionen.

Formuliere einen gemeinsamen Vertrag für lineare und binäre Suche. Nur die innere Strategie unterscheidet sich. Die Suche wird vollständig ausprogrammiert; eingebaute Suchhilfen wie Python `list.index()` oder PHP `array_search()` sind in dieser Lern- und Prüfungsaufgabe nicht zulässig.

Festlegung	Vereinbarung im Teammanager
Eingabe	Array kader und exakt geschriebener Suchname
Erfolg	Index des ersten passenden Elements
Misserfolg	Rückgabewert -1

Festlegung	Vereinbarung im Teammanager
Leeres Array	ebenfalls –1; kein ungültiger Indexzugriff
Doppelte Namen	die lineare Suche liefert den ersten Treffer; IDs wären langfristig eindeutiger

WICHTIG

Index und menschlich gelesene Position sind nicht dasselbe: Index 0 entspricht Position 1. Der Rückgabewert –1 darf deshalb nicht ungeprüft als Arrayindex verwendet werden.

5.2 Lineare Suche: Element für Element

Struktogramm einer linearen Suche

BW-Operatorenliste 2.2 · Ablauf prüfen und Rückgabe bestimmen



Prüfauftrag: Bestimme die Rückgabe für `gesucht = 8` und `gesucht = 6`.

Abbildung 4: Vorgegebenes Struktogramm einer linearen Suche.

Abiturtraining: Führe das Struktogramm für `werte = [3, 5, 8, 11]` einmal mit `gesucht = 8` und einmal mit `gesucht = 6` aus. Notiere die geprüften Indizes und die jeweilige Rückgabe.

Die lineare Suche beginnt an Index 0 und prüft nacheinander jedes Element. Sie benötigt keine Sortierung. Sobald ein Treffer gefunden ist, endet die Funktion; andernfalls liefert sie nach dem letzten Element -1 .

Vergleich	Index	Arraywert	Ergebnis
1	0	Mina	nicht Daria
2	1	Ben	nicht Daria
3	2	Aylin	nicht Daria
4	3	Cem	nicht Daria
5	4	Daria	gefunden: Index 4

Python	PHP
<pre> def lineare_suche(kader, gesucht): for index in range(len(kader)): if kader[index] == gesucht: return index return -1 kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"] gesucht = input("Spielername: ").strip() index = lineare_suche(kader, gesucht) if index == -1: print("Nicht gefunden") else: print("Gefunden an Position", index + 1) </pre>	<pre> <?php function lineareSuche(array \$kader, string \$gesucht): int { for (\$index = 0; \$index < count(\$kader); \$index++) { if (\$kader[\$index] === \$gesucht) { return \$index; } } return -1; } \$kader = ["Mina", "Ben", "Aylin", "Cem", "Daria"]; \$gesucht = trim(readline("Spielername: ")); \$index = lineareSuche(\$kader, \$gesucht); if (\$index === -1) { echo "Nicht gefunden"; } else { echo "Gefunden an Position " . (\$index + 1); } </pre>

HILFESTELLUNG

Die Schleife prüft nur gültige Indizes. return beendet die Funktion beim ersten Treffer. Das abschließende return -1 wird nur erreicht, wenn kein Element übereinstimmt. Teste den ersten, einen mittleren, den letzten und keinen Treffer.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich arbeite an der linearen Suche. Gesuchter Wert: [einsetzen]. Mein bisher geprüfter Bereich: [einsetzen]. Frage mich zuerst, welcher Index als Nächstes gültig ist und wodurch Treffer und Nichttreffer unterschieden werden. Gib nur eine Hilfestufe auf einmal und keine vollständige Schleife. Wenn du ein Beispiel brauchst, verwende ein anderes Array. Lass mich selbst bestimmen, an welcher Stelle return stehen muss, und fordere danach einen Test für ersten, mittleren, letzten und fehlenden Wert. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

5.3 Binäre Suche: den Suchbereich halbieren

Binäre Suche

Sortierte Liste · Suchbereich wird halbiert



Abbildung 5: Erster Prüfschritt der binären Suche nach 31.

Führe die Suche bis zum Treffer fort und protokolliere links, rechts und mitte nach jedem Schritt.

Die binäre Suche nutzt die Sortierung des Arrays. Sie vergleicht den Suchwert mit dem mittleren Element und verwirft anschließend die Hälfte, in der der Wert nicht liegen kann. Der Suchbereich wird durch start und ende einschließlich begrenzt.

Schritt	start	mitte	ende	Vergleich mit Daria	Folge
1	0	2	4	Cem < Daria	start = 3
2	3	3	4	Daria = Daria	gefunden: Index 3

Python	PHP
<pre>def binaere_suche(kader, gesucht): start = 0 ende = len(kader) - 1 while start <= ende: mitte = (start + ende) // 2 if kader[mitte] == gesucht: return mitte if kader[mitte] < gesucht: start = mitte + 1 else: ende = mitte - 1 return -1 kader = ["Aylin", "Ben", "Cem", "Daria", "Mina"] gesucht = input("Spielernamen: ") print(binaere_suche(kader, gesucht))</pre>	<pre><?php function binaereSuche(array \$kader, string \$gesucht): int { \$start = 0; \$ende = count(\$kader) - 1; while (\$start <= \$ende) { \$mitte = intval(\$start + \$ende, 2); if (\$kader[\$mitte] === \$gesucht) { return \$mitte; } if (\$kader[\$mitte] < \$gesucht) { \$start = \$mitte + 1; } else { \$ende = \$mitte - 1; } } return -1; } \$kader = ["Aylin", "Ben", "Cem", "Daria", "Mina"]; \$gesucht = trim(readline("Spielernamen: ")); echo binaereSuche(\$kader, \$gesucht);</pre>

FEHLERQUELLE

Wird ein unsortiertes Array binär durchsucht, kann ein vorhandener Wert fälschlich ausgeschlossen werden. Sortiert bedeutet außerdem: Suche und Sortierung müssen dieselbe Vergleichsregel verwenden, etwa Groß-/Kleinschreibung und Umlaute betreffend.

ARBEITSAUFTRAG

Führe die binäre Suche als Schreibtischtest für Aylin, Daria, Mina und Zoe aus. Protokolliere start, mitte, ende, den verglichenen Wert und die Entscheidung. Modellierte den Ablauf anschließend mit „Wiederhole solange ...“, dem Wenn-Operator, Zuweisungen, Arrayzugriff und „Rückgabe: wert“ nach Operatorenliste Version 2.2. Erkläre, warum start $\leq$ ende auch ein Array mit nur einem verbleibenden Element prüft.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich führe eine binäre Suche im sortierten Array [einsetzen] nach [Suchwert] aus. Meine aktuellen Grenzen sind start = [], mitte = [], ende = []. Prüfe zuerst nur, ob meine Mitte und mein Vergleich korrekt sind. Frage mich anschließend, welche Hälfte sicher ausgeschlossen werden kann und warum. Verrate nicht die neuen Grenzen, bevor ich sie selbst genannt habe. Gib bei Bedarf erst eine Denkfrage, dann einen Strukturhinweis und zuletzt ein Teilbeispiel mit anderen Zahlen. Liefere keine fertige Tabelle und keinen vollständigen Code. Prüfe am Ende mit mir den Nicht-gefunden-Fall. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

5.4 Lineare und binäre Suche vergleichen

Kriterium	Lineare Suche	Binäre Suche
Voraussetzung	keine Sortierung erforderlich	Array muss passend sortiert sein
Schritt	nächstes Element prüfen	mittleres Element prüfen, Hälfte verwerfen
Schlechtester Fall	bis zu n Vergleiche	Suchbereich wird wiederholt halbiert
Geeignet	kleine oder häufig veränderte unsortierte Liste	große, bereits sortierte und häufig durchsuchte Liste
Fehlerfall	-1 nach letztem Element	-1 , wenn start größer als ende wird

ZUSAMMENHANG

Die binäre Suche zeigt, warum Sortieren und Suchen im Lehrplan zusammengehören. Die einmalige Sortierung verursacht Aufwand, kann aber viele spätere Suchvorgänge beschleunigen. Bei nur einer Suche kann eine vorherige Sortierung dagegen unnötig sein.

Kapitelaufgabe: Spiellersuche im Teammanager · ADS-K05-A01

ARBEITSAUFTRAG

Das Hauptprogramm liest einen Suchnamen über die Shell ein und ruft die Suchfunktionen auf. Suche zuerst linear in der unsortierten Trainingsliste. Sortiere anschließend eine Kopie mit Selection Sort oder Bubble Sort und suche darin binär. Gib Position, Zahl der Vergleiche und Nicht-gefunden-Fall verständlich aus. Verwende für beide Sprachen dieselben Testdaten.

Abgabeprodukte: DS05_lineare_suche und DS05_binaere_suche als Struktogrammer-Originaldateien mit SVG/PNG-Export, DS05_suchen.py; im BKWI1 zusätzlich DS05_suchen.php, Zustandsprotokolle und Testtabelle.

HILFESTELLUNG

Basis: Ergänze Rückgabewert und Nicht-gefunden-Ausgabe in vorgegebenem Code. Standard: Implementiere beide Suchverfahren und protokolliere ihre Zustände. Plus: Zähle Vergleiche bei unterschiedlich großen Arrays und begründe, wann sich vorheriges Sortieren lohnt.

AUSBlick LEARNING AGENT

Der Learning Agent fordert vor jedem Schritt eine Prognose an: nächster Index bei linearer Suche oder neue Grenzen bei binärer Suche. Hinweise werden gestuft gegeben; die vollständige Lösung erscheint erst nach einem eigenen Versuch.

SELBSTKONTROLLE

Ich kann den Suchvertrag und -1 erklären; lineare Suche auf beliebigen Arrays ausführen; die Sortier-Voraussetzung der binären Suche begründen; start, mitte und ende korrekt aktualisieren; Erfolg und Misserfolg testen; beide Verfahren passend auswählen.

Kapitel 6: Persistenz – JSON und relationale Datenbank ↩ Inhalt

KAPITELZIEL

Du unterscheidest flüchtige und dauerhafte Datenhaltung, modellierst denselben fachlichen Sachverhalt redundanzarm für JSON und für eine relationale Datenbank und greifst in Python und PHP darauf zu. Damit verbindet dieses Kapitel Programmierung, das relationale Datenbank-Skript und die spätere objektorientierte Entwicklung.

Ausgangssituation: Der Learning Agent soll sich erinnern

Während ein Programm läuft, verwaltet es Lernende, Aufgaben und Bearbeitungsversuche in Listen beziehungsweise Arrays. Nach Programmende sind diese Werte verloren. Der Learning Agent soll beim nächsten Start aber weiterhin wissen, welche Aufgabe bearbeitet wurde, wie viele Punkte erreicht wurden und wann der Versuch stattfand.

ARBEITSAUFTRAG

Notiere zunächst ohne Quellcode: Welche Objekte kommen vor? Welche eindeutigen IDs benötigen sie? Welche Beziehungen bestehen? Welche Daten ändern sich häufig? Welche Abfragen soll eine Lehrkraft stellen können?

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich entwerfe die Datenhaltung für [Anwendungsfall]. Meine bisher erkannten Objekte und Attribute sind: [einsetzen]. Stelle mir nacheinander Fragen zu eindeutigen IDs, Beziehungen, mehrfach vorkommenden Daten und typischen Abfragen. Nenne keine fertigen Tabellen und kein fertiges JSON-Schema. Weise nur auf eine mögliche Redundanz oder fehlende Beziehung hin und warte auf meine Überarbeitung. Lass mich am Ende begründen, welche Daten getrennt gespeichert und über IDs verbunden werden. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

DIFFERENZIERUNG

BG: Datenmodell lesen, Beziehungen erklären und JSON mit einem relationalen Schema vergleichen. BKWI1: zusätzlich JSON-Datei und SQLite-Datenbank in Python und PHP praktisch anbinden; anschließend den Transfer zu MySQL Workbench herstellen.

6.1 Flüchtig oder persistent?

Lernfortschritt

Persistenz als Dienst hinter einer Oberfläche

Meine Aufgaben

Schleifen	8 / 10	gespeichert
Sortieren	6 / 10	gespeichert
Suchen	9 / 10	gespeichert

Neuen Versuch speichern

Abbildung 11: Lernfortschrittsoberfläche vor der Wahl der Speicherung.

Trenne die Oberfläche von den Funktionen laden, versuch_hinzufuegen und speichern. Welche Rückgaben sind für unabhängige Tests sinnvoll?

Speicherform	Geeignet für	Wichtige Grenze
Liste / Array im Arbeitsspeicher	Berechnungen und Zustände während eines Programmlaufs	Inhalt geht beim Beenden verloren.
JSON-Datei	Kleine Datenmengen, Austausch, Konfiguration und Prototypen	Beziehungen, gleichzeitige Zugriffe und Konsistenz muss die Anwendung weitgehend selbst sichern.
Relationale Datenbank	Dauerhafte, strukturierte und gemeinsam genutzte Daten mit gezielten Abfragen	Schema, Schlüssel, Zugriffslogik und Betrieb müssen geplant werden.

MERKSATZ

JSON ist kein allgemeiner Ersatz für eine relationale Datenbank. Entscheidend sind Datenmenge, Beziehungen, Abfragen, Mehrbenutzerbetrieb und die geforderte Konsistenz.

6.2 Ein fachliches Modell - zwei Speicherformen

Für den Learning Agent werden drei fachliche Mengen getrennt. Ein Bearbeitungsversuch speichert nicht noch einmal Namen und Aufgabentitel, sondern verweist über IDs auf Lernende und Aufgaben.

Relation / Sammlung	Attribute	Bedeutung
lernende	lernende_id, name	Eine Person wird über lernende_id eindeutig identifiziert.
aufgaben	aufgabe_id, titel, max_punkte	Eine Aufgabe wird einmal beschrieben.
versuche	versuch_id, lernende_id, aufgabe_id, punkte, zeitpunkt	Der Versuch verbindet eine Person mit einer Aufgabe.

FACHLICH GENAU

Die dritte Normalform (3NF) ist ein formaler Begriff für relationale Schemata. Eine JSON-Datei ist daher nicht „in 3NF“. Sie kann aber dasselbe Ziel unterstützen: wiederholte Stammdaten vermeiden, Informationen nach fachlichen Objekten trennen und Beziehungen durch IDs ausdrücken.

Beispiel einer redundanzarmen JSON-Struktur:

```
{
  "lernende": [{"lernende_id": 1, "name": "Mina"}],
  "aufgaben": [{"aufgabe_id": 12, "titel": "Queue erklären",
    "max_punkte": 10}],
  "versuche": [{"versuch_id": 101, "lernende_id": 1,
    "aufgabe_id": 12, "punkte": 8,
    "zeitpunkt": "2026-09-06T10:15:00"}]
}
```

SELBSTKONTROLLE

Erkläre: Warum stehen name und titel nicht zusätzlich in jedem Versuch?
Was müsste beim Ändern eines Namens geschehen, wenn diese Werte vielfach kopiert wären?

6.3 JSON lesen, verändern und speichern

Beide Programme lesen dieselbe Datei `lerndaten.json`, ergänzen einen Versuch und schreiben den gesamten Datenbestand zurück. Die Schlüssel der JSON-Objekte bleiben in beiden Sprachen gleich.

Python	PHP
<pre>import json from pathlib import Path def daten_laden(pfad): return json.loads(Path(pfad).read_text(encoding="utf-8")) print(daten_laden("lerndaten.json"))</pre>	<pre><?php function datenLaden(\$pfad) { return json_decode(file_get_contents(\$pfad), true, 512, JSON_THROW_ON_ERROR); } print_r(datenLaden(__DIR__ . "/lerndaten.json"));</pre>

HILFESTELLUNG

Python: `json.loads` wandelt Text in Listen und Dictionaries um; `json.dumps` erledigt den Rückweg. PHP: `json_decode(..., true)` liefert verschachtelte Arrays; `json_encode` erzeugt wieder JSON. Prüfe vor produktivem Einsatz zusätzlich Datei- und Formatfehler.

ARBEITSAUFTRAG

Ergänze zwei Lernende, zwei Aufgaben und insgesamt vier Versuche. Gib für jeden Versuch Name, Aufgabentitel und Punkte aus. Dazu musst du die passenden IDs in den drei Sammlungen zusammenführen.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich lese verschachtelte JSON-Daten ein und muss Lernende, Aufgaben und Versuche über IDs zusammenführen. Mein Zwischenergebnis ist: [einsetzen]. Frage mich zuerst, in welcher Sammlung die benötigte Information liegt und welche ID die Verbindung herstellt. Gib danach nur einen Hinweis zur nächsten Such- oder Schleifenoperation. Schreibe keinen vollständigen Code und keine vollständige Ausgabe. Lass mich selbst einen fehlenden Verweis als Fehlerfall erkennen und behandeln. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

6.4 Vom JSON-Modell zur relationalen Datenbank

SQLite eignet sich für den unmittelbaren Unterrichtseinsatz: Die Datenbank liegt in einer Datei und benötigt keinen separaten Server. Das relationale Modell bleibt übertragbar. Im Datenbankunterricht werden dieselben Tabellen, Primär- und Fremdschlüssel sowie JOIN-Abfragen im vorhandenen Skript vertieft und mit MySQL Workbench umgesetzt.

Relationales Element	Realisierung	Nutzen
Primärschlüssel	lernende_id, aufgabe_id, versuch_id	Datensätze eindeutig ansprechen
Fremdschlüssel	versuche.lernende_id und versuche.aufgabe_id	Beziehungen sichern
Constraints	NOT NULL, CHECK, FOREIGN KEY	Ungültige Daten früh verhindern

Relationales Element	Realisierung	Nutzen
JOIN	versuche mit lernende und aufgaben verbinden	Fachlich lesbare Ergebnisse erzeugen

```

CREATE TABLE lernende (
  lernende_id INTEGER PRIMARY KEY,
  name TEXT NOT NULL
);
CREATE TABLE aufgaben (
  aufgabe_id INTEGER PRIMARY KEY,
  titel TEXT NOT NULL,
  max_punkte INTEGER NOT NULL CHECK (max_punkte > 0)
);
CREATE TABLE versuche (
  versuch_id INTEGER PRIMARY KEY,
  lernende_id INTEGER NOT NULL REFERENCES lernende(lernende_id),
  aufgabe_id INTEGER NOT NULL REFERENCES aufgaben(aufgabe_id),
  punkte INTEGER NOT NULL CHECK (punkte >= 0),
  zeitpunkt TEXT NOT NULL
);

```

Vorbereitet einfügen: Python und PHP

Python	PHP
<pre> import sqlite3 def versuche_laden(pfad): with sqlite3.connect(pfad) as db: db.execute("PRAGMA foreign_keys = ON") return db.execute("SELECT * FROM versuch").fetchall() print(versuche_laden("lerndaten.db")) </pre>	<pre> <?php function versucheLaden(\$pfad) { \$db = new PDO("sqlite:" . \$pfad); \$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); return \$db->query("SELECT * FROM versuch")->fetchAll(PDO::FETCH_ASSOC); } print_r(versucheLaden("lerndaten.db")); </pre>

SICHERER ZUGRIFF

Werte werden als Parameter übergeben und nicht in den SQL-Text zusammengesetzt. Transaktionen fassen zusammengehörige Änderungen zusammen; Constraints schützen Regeln auch dann, wenn eine Anwendung einen Fehler enthält.

ARBEITSAUFTRAG

Überführe die Beispieldaten in SQLite. Formuliere anschließend einen JOIN, der Name, Aufgabentitel, erreichte Punkte und Zeitpunkt ausgibt. Öffne das Schema im BKWI zusätzlich in MySQL Workbench und kennzeichne notwendige Dialektänderungen.

WENN DU FESTHÄNGST · ZUSÄTZLICHER DENKPROMPT

Ich formuliere einen JOIN zwischen [Tabellen einsetzen]. Gewünschte Ausgabefelder: [einsetzen]. Frage mich zuerst, aus welcher Tabelle jedes Feld stammt und über welche Primär- und Fremdschlüssel die Tabellen verbunden sind. Gib pro Schritt nur eine Frage oder einen Strukturhinweis. Schreibe nicht die vollständige SQL-Abfrage. Wenn nötig, zeige nur ein JOIN-Teilstück mit anderen Tabellennamen. Lass mich anschließend erklären, wodurch unerwünschte Mehrfachzeilen oder fehlende Treffer entstehen können. Beende den Dialog mit einem kurzen Lernweg-Protokoll: Ausgangsidee – konkrete Hürde – genutzte Hinweisstufe – eigene Änderung – Test/Ergebnis – offene Frage. Übernimm nicht den gesamten KI-Chat.

6.5 Brücke zur objektorientierten Programmierung

Persistenz soll nicht über die gesamte Anwendung verteilt werden. Eine klare Aufgabenteilung erleichtert den späteren Wechsel von JSON zu einer Datenbank.

Baustein	Verantwortung	Mögliches Beispiel
Fachobjekte	Zustand eines Gegenstands modellieren	Lernender, Aufgabe, Versuch
Service / Geschäftslogik	Regeln prüfen und Abläufe steuern	VersuchBewertenService
Repository / Datenzugriff	Laden und Speichern kapseln	JsonVersuchRepository, SqlVersuchRepository
Benutzungsoberfläche	Eingaben aufnehmen und Ergebnisse darstellen	Shell, Web-Interface, spätere Lernplattform

MERKSATZ

Fachobjekte modellieren die Domäne. Services realisieren Regeln. Repositories kapseln die Persistenz. Die Oberfläche führt den Dialog. So kann dieselbe Business-Logik später von der Shell in eine Lernplattform übernommen werden.

AUSBLICK LEARNING AGENT

Der Learning Agent kann Versuche zunächst aus JSON laden. Später greift ein Datenbank-Repository auf dieselben fachlichen Objekte zu. Eine Web-Oberfläche oder E-Learning-Plattform ruft die Geschäftslogik auf, ohne Speicherformat und SQL kennen zu müssen.

6.6 Optionaler Ausblick: Lernwege für die Plattform erfassen

Neben Punkten kann eine spätere Lernplattform ausgewählte Denkschritte strukturiert speichern. Dabei wird nicht der vollständige KI-Dialog gesammelt. Ein knapper Lernwegeintrag verbindet Aufgabe, Bearbeitungsversuch, Hürde, verwendete Hilfestufe, eigene Änderung und Prüfung.

Feld	Beispiel / Bedeutung
lernweg_id	eindeutige ID des protokollierten Lernwegs
versuch_id	Verbindung zum zugehörigen Bearbeitungsversuch
huerde_kategorie	zum Beispiel BEDINGUNG, INDEX oder TESTFALL
ausgangsidee	kurze Zusammenfassung in eigenen Worten
denkprompt	die tatsächlich gestellte Lernfrage
hinweisstufe	0 bis 3; niedrigste ausreichende Unterstützung
eigene_aenderung	der danach selbst ausgeführte Schritt
pruefung_ergebnis	Testfall, Ergebnis und noch offene Frage

AUSWERTUNG STATT ÜBERWACHUNG

Die Lehrkraft wertet Muster aus, nicht Persönlichkeiten: häufige Hürdenkategorien, wirksame Diagnosefragen, zu frühe Lösungsausgaben und typische Fehlvorstellungen.

Personenbezogene Vergleiche oder eine automatische Benotung des Denkwegs sind nicht das Ziel. Bevor echte Schülerdaten gespeichert werden, müssen schulische Datenschutzvorgaben, Zugriffsrechte und Löschfristen verbindlich geklärt sein.

ARBEITSAUFTRAG

Erweitere das relationale Modell optional um lernwege. Verbinde jeden Lernwegeintrag über `versuch_id` mit genau einem Versuch. Formuliere anschließend zwei anonymisierte Auswertungsfragen: 1. Welche Hürdenkategorie kommt häufig vor? 2. Welche niedrigste Hinweisstufe führte meistens zu einem erfolgreichen eigenen Test?

Kapitelaufgabe: Lernfortschritt dauerhaft speichern · ADS-K06-A01

ARBEITSAUFTRAG

1. Zeichne Lernende, Aufgaben und Versuche mit IDs und Beziehungen.
2. Erzeuge `lerndaten.json` mit zwei Lernenden, zwei Aufgaben und vier Versuchen.
3. Lies und ergänze die Datei.
4. Erzeuge `schema.sql` und übertrage die Daten nach `lerndaten.db`.
5. Gib mit einem JOIN die fachlich verständliche Übersicht aus.
6. Vergleiche JSON und Datenbank anhand von Redundanz, Beziehungen, Abfragen und Änderbarkeit.

Abgabeprodukte: `DS06_persistenzmodell` (drawio oder SVG), `lerndaten.json`, `schema.sql`, `lerndaten.db`, `DS06_persistenz.py`; im BKWI1 zusätzlich `DS06_persistenz.php` sowie eine Vergleichs- und Testtabelle.

HILFESTELLUNG

Basis: Nutze die vorgegebenen Tabellen und ergänze nur Beispieldaten und Ausgabe. Standard: Entwickle JSON- und SQLite-Lösung vollständig. Plus: Definiere eine gemeinsame Repository-Schnittstelle und tausche die Speicherung aus, ohne die Geschäftslogik zu verändern.

SELBSTKONTROLLE

Ich kann erklären, warum Arbeitsspeicher nicht persistent ist; JSON und relationale Datenbank begründet auswählen; Primär- und Fremdschlüssel zuordnen; einen vorbereiteten INSERT und einen JOIN erläutern; die Rolle einer Datenzugriffsschicht für OOP und Lernplattform beschreiben.

Zwischenstand nach Kapitel 6

DER GEMEINSAME FADEN

Vom einfachen Array über Operationen, Sortieren und Suchen bis zur persistenten Speicherung bleibt die fachliche Aufgabe erkennbar: Daten passend strukturieren, Business-Logik sprachneutral planen, in Python und PHP umsetzen und mit nachvollziehbaren Testfällen prüfen.

Was du bis hierhin verbinden kannst

1. Datenstruktur nach Zugriffsregel und benötigten Operationen auswählen.
2. Selection Sort und Bubble Sort auf eindimensionalen Arrays nachvollziehen und implementieren.
3. Lineare und binäre Suche passend zur Ausgangslage auswählen und sicher implementieren.
4. Daten aus dem Arbeitsspeicher redundanzarm in JSON oder relationalen Tabellen persistieren.

NÄCHSTER ENTWICKLUNGSSCHRITT

Die fachlichen Funktionen können im folgenden Schuljahr in Klassen, Services und Repositories überführt werden. Eine spätere Lernplattform nutzt dieselbe Geschäftslogik und ergänzt Web-Oberfläche, Benutzerverwaltung und dauerhafte Datenhaltung.

QUELLENBASIS

Repository python-algorithmen-datenstrukturen; BPE 7 und Niveauplan; Abiturprüfung Baden-Württemberg: Operatorenliste für Struktogramme, Version 2.2 vom 01.09.2024; L2- Informationsdokumente zu Algorithmen, Selection Sort, Bubble Sort, linearer und binärer Suche; L3-Informationsdokumente; Volleyball-Struktogramme; BG-Lehrplan Informatik Abitur 2021; vorhandenes Skript zu relationalen Datenbanken; offizielle Referenzen zu Python collections.deque, json und sqlite3 sowie PHP SplQueue, JSON und PDO_SQLite.

KI-HINWEIS

Dieses Dokument wurde mit Unterstützung künstlicher Intelligenz erstellt und fachlich redaktionell weiterbearbeitet. Trotz sorgfältiger Prüfung können Fehler oder Unvollständigkeiten enthalten sein. Hinweise und Korrekturen bitte an die Autorin melden.

Kapitel 7: Verkettete Liste

Warum brauchst du das später?

Dynamische Folgen speichern Elemente in Knoten; jeder Knoten verweist auf seinen Nachfolger. Das Modell macht Referenzen und Einfügen ohne Verschieben sichtbar.

DARSTELLUNG · erst erklären, dann programmieren

Beispiel: $\text{START} \rightarrow [A \mid \bullet] \rightarrow [B \mid \bullet] \rightarrow [C \mid \emptyset]$. Der linke Zellteil enthält den Wert, der rechte den Verweis. Einfügen zwischen A und B: neuer Knoten zeigt auf B, danach zeigt A auf den neuen Knoten.

7.1 Struktur lesen und zeichnen

Playlist bearbeiten

Verkettete Liste als sichtbare Folge

Wiedergabeliste

01 Start	:	Davor einfügen
02 Training	:	Danach einfügen
03 Abschluss	:	Entfernen

Abbildung 12: Playlist als Benutzeroberfläche einer verketteten Liste.

Ordne jeder Schaltfläche eine Methode zu. Zeichne zuerst die Zeigeränderung und teste die Methode anschließend durch einen separaten Aufruf.

Arbeite zuerst mit Kästchen, Pfeilen und einem Zustandsprotokoll. Nutze für prüfungsnahе Struktogramme den Struktogrammer aus der Digitalen Schultasche. Draw.io ist nur eine ergänzende Lernhilfe und in Klassenarbeiten nicht zulässig.

DENKCOACH-PROMPT

Ich habe die Knoten $A \rightarrow B \rightarrow C$. Stelle mir nacheinander höchstens drei Fragen, damit ich selbst herausfinde, welche zwei Verweise beim Einfügen von X zwischen A und B geändert werden. Zeige keinen fertigen Code.

Gestufte Tutorhilfe

Stufe 1: Lass dir nur eine Gegenfrage stellen. Stufe 2: Bitte um einen Hinweis auf die nächste Operation. Stufe 3: Bitte um ein ähnliches Mini-Beispiel mit anderen Daten. Verlange ausdrücklich keine fertige Lösung.

Prüfnachweis · ADS-K07-A01

AUFGABE

Zeichne die Liste $K1 \rightarrow K2 \rightarrow K3$. Füge KX an Position 2 ein und protokolliere die Zeigeränderungen. BKWI-Erweiterung: implementiere `append`, `insert_after` und `remove` in Python oder PHP.

Abnahme: Zeichnung vor/nachher; keine verlorenen Knoten; mindestens Normalfall und Randfall getestet.

Vertiefung · optional für schnelle Schüler

Vergleiche Aufwand und Fehlerrisiko beim Einfügen mit einem Array.

Transfer zum Learning Agent

Notiere Ausgangsidee, verwendeten Denkprompt, Antwort in eigenen Worten, Änderung am Modell und Testbeleg im Lernwegprotokoll. So bleibt der eigene Denkweg auswertbar.

Mini-Check

Ich kann die Struktur zeichnen, die erlaubten Operationen benennen, einen Zustandsschritt erklären und einen passenden Testfall begründen.

Kapitel 8: Stapelspeicher (Stack)

Warum brauchst du das später?

Ein Stack modelliert LIFO: Das zuletzt abgelegte Element wird zuerst entnommen. Praxisbeispiele sind Rückgängig-Funktionen, Browser-Verlauf und verschachtelte Aufrufe.

DARSTELLUNG · erst erklären, dann programmieren

Beispiel mit TOP rechts: [Startseite, Produktseite, Warenkorb] ← TOP. push(Checkout) legt rechts ab; pop() liefert Checkout; peek() liest Warenkorb ohne Entnahme.

8.1 Struktur lesen und zeichnen

Browser-Verlauf als Stack

LIFO: zuletzt besucht – zuerst zurück

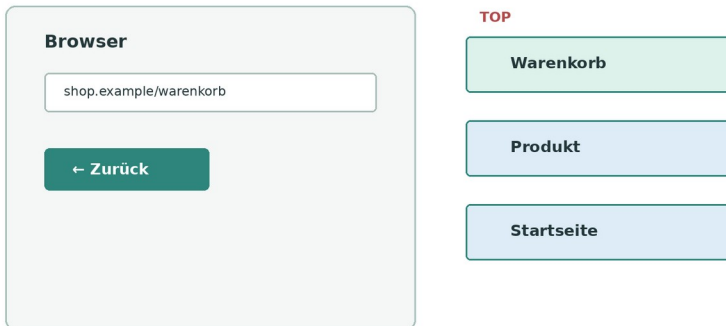


Abbildung 6: Browser-Zurück-Funktion als Stack.

Welche Seite erscheint nach einem pop? Ergänze anschließend zwei Besuche mit push und zeichne den neuen Stack.

Arbeite zuerst mit Kästchen, Pfeilen und einem Zustandsprotokoll. Nutze für prüfungsnahе Struktogramme den Struktogrammer aus der Digitalen Schultasche. Draw.io ist nur eine ergänzende Lernhilfe und in Klassenarbeiten nicht zulässig.

DENKCOACH-PROMPT

Führe mich mit Fragen durch die Zustände eines Stacks. Gib mir nach jedem meiner Schritte nur „plausibel“ oder einen Hinweis auf die verletzte LIFO-Regel. Keine Gesamtlösung.

Gestufte Tutorhilfe

Stufe 1: Lass dir nur eine Gegenfrage stellen. Stufe 2: Bitte um einen Hinweis auf die nächste Operation. Stufe 3: Bitte um ein ähnliches Mini-Beispiel mit anderen Daten. Verlange ausdrücklich keine fertige Lösung.

Prüfnachweis · ADS-K08-A01**AUFGABE**

Modelliere einen Browser-Verlauf mit push, pop und peek. Erstelle ein Zustandsprotokoll für sechs vorgegebene Operationen. BKWI: implementiere den Stack mit Fehlerbehandlung für einen leeren Speicher.

Abnahme: Rückgabewerte und TOP nach jedem Schritt korrekt; Unterlauf wird behandelt; zwei eigene Testfälle.

Vertiefung · optional für schnelle Schüler

Prüfe Klammerausdrücke mit einem Stack.

Transfer zum Learning Agent

Notiere Ausgangsidee, verwendeten Denkprompt, Antwort in eigenen Worten, Änderung am Modell und Testbeleg im Lernwegprotokoll. So bleibt der eigene Denkweg auswertbar.

Mini-Check

Ich kann die Struktur zeichnen, die erlaubten Operationen benennen, einen Zustandsschritt erklären und einen passenden Testfall begründen.

Kapitel 9: Binärer Suchbaum

Warum brauchst du das später?

Ein binärer Suchbaum ordnet kleinere Schlüssel links und größere rechts. Das schafft eine Brücke zu rekursiven Algorithmen und späteren Datenbankindizes.

DARSTELLUNG · erst erklären, dann programmieren

Beispiel: Wurzel 50; links 30 mit Kindern 20 und 40; rechts 70 mit Kindern 60 und 80. Inorder-Durchlauf ergibt 20, 30, 40, 50, 60, 70, 80.

9.1 Struktur lesen und zeichnen

Suchbaum prüfen

Aufruf, Suchpfad und Rückgabe sichtbar machen



Schlüssel

contains testen

Suchpfad: 50 → 70 → 60 → leer

Rückgabe: falsch

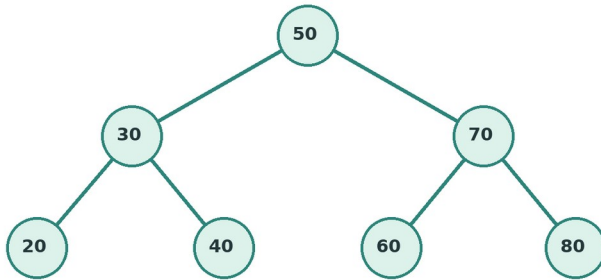
Jede Entscheidung wird mit kleiner/größer begründet.

Abbildung 13: Suchbaumoberfläche mit sichtbarem Suchpfad und Rückgabewert.

Die Methode contains erhält Wurzel und Schlüssel. Protokolliere ihren Aufruf für Treffer und Nichttreffer und vergleiche mit dem Struktogramm.

Binärer Suchbaum

Kleinere Schlüssel links · größere Schlüssel rechts

**Prüffrage: Welchen Suchpfad nimmt 60? Und 65?****Abbildung 7: Binärer Suchbaum für Suchpfadaufgaben.**

Protokolliere die Suchpfade für 60 und 65. Begründe an jedem Knoten die Entscheidung links oder rechts.

Arbeite zuerst mit Kästchen, Pfeilen und einem Zustandsprotokoll. Nutze für prüfungsnahе Struktogramme den Struktogrammer aus der Digitalen Schultasche. Draw.io ist nur eine ergänzende Lernhilfe und in Klassenarbeiten nicht zulässig.

DENKCOACH-PROMPT

Gib mir die Einfügefolge 50, 30, 70, 20, 40. Frage mich für jeden neuen Wert, ob ich links oder rechts gehe, und warte auf meine Begründung. Zeichne den fertigen Baum erst nach meiner eigenen Skizze.

Gestufte Tutorhilfe

Stufe 1: Lass dir nur eine Gegenfrage stellen. Stufe 2: Bitte um einen Hinweis auf die nächste Operation. Stufe 3: Bitte um ein ähnliches Mini-Beispiel mit anderen Daten. Verlange ausdrücklich keine fertige Lösung.

Prüfnachweis · ADS-K09-A01

AUFGABE

Erzeuge aus 50, 30, 70, 20, 40, 60, 80 einen Suchbaum.
Protokolliere die Suche nach 60 und 65. BKWI-Erweiterung:
implementiere insert und contains.

Abnahme: Suchpfade nachvollziehbar; Ordnungsregel an jedem Knoten erfüllt; Treffer und Nichttreffer getestet.

Vertiefung · optional für schnelle Schüler

Untersuche, wie die Einfügefølge die Baumhöhe beeinflusst.

Transfer zum Learning Agent

Notiere Ausgangsidee, verwendeten Denkprompt, Antwort in eigenen Worten, Änderung am Modell und Testbeleg im Lernwegprotokoll. So bleibt der eigene Denkweg auswertbar.

Mini-Check

Ich kann die Struktur zeichnen, die erlaubten Operationen benennen, einen Zustandsschritt erklären und einen passenden Testfall begründen.

Abschluss und Ausblick

Du kannst nun Datenstrukturen darstellen und auswählen, Listen verarbeiten, Sortier- und Suchalgorithmen nachvollziehen sowie Persistenz einordnen. Die Vertiefung führt zur objektorientierten Modellierung und zur späteren Lernplattform mit dokumentierten Lernwegen.

Hinweise, Quellen und Lizenz

1.5 · Arbeitsfassung 2026/27 · 10.09.2026

Custom License – Christine Janischek NC School-Only 1.0. Nicht-kommerzielle Nutzung ausschließlich im staatlichen Schulwesen.

Sichtbare Namensnennung: Christine Janischek –

<https://emotionalspirit.de>. Andere Nutzungen nur mit Genehmigung.

KI-Hinweis: Bei Konzeption, Formulierung und technischer Prüfung wurde KI unterstützend eingesetzt. Trotz Prüfung können Fehler verbleiben. Bitte Fehler und Verbesserungsvorschläge an die Autorin melden.

Amtliche Referenz: Operatorenliste für Struktogramme, Version 2.2, Schule Baden-Württemberg (01.09.2024). Ausgangs- und Erprobungsmaterial: <https://github.com/ChristineJanischek/python-algorithmen-datenstrukturen>

Prüfstatus: barrierearm. Automatisierte Struktur- und Layoutprüfungen sind erfolgt; manueller Screenreader- und Fremdrechnertest bleiben vor einer Aussage „barrierefrei geprüft“ offen.