



Schulcurriculum Informatik

Jahrgangsstufen 11 – 12 (Oberstufe)

Madrid, Juli 2025

Gültig ab dem Schuljahr 2025/26

Deutsche Schule Madrid, Fachschaft Informatik
Monasterio de Guadalupe, 7, 28049 Madrid
Tel.: +34 915 580 200
info@dsmadrid.org
www.dsmadrid.org

Das vorliegende Schulcurriculum wurde vom Bund-Länder-Ausschuss für schulische Arbeit im Ausland im September 2025 genehmigt.

Vorwort

Unsere Lebens- und Arbeitswelt ist zunehmend durch digitale Technologien, Automatisierung, Künstliche Intelligenz und datenbasierte Systeme geprägt. Das überarbeitete Schulcurriculum Informatik der Deutschen Schule Madrid (DSM) reagiert auf diese Veränderungen und bildet die Grundlage für einen modernen, zukunftsorientierten Informatikunterricht in der Sekundarstufe II.

Das vorliegende Schulcurriculum für das Fach Informatik an der Deutschen Schule Madrid stellt eine umfassende Überarbeitung bzw. Neugestaltung des im Schuljahr 2021/22 implementierten DSM-Informatik-Lehrplans dar. Es wurde in zentralen Bereichen aktualisiert und erweitert, um der rasanten technologischen Entwicklung der vergangenen Jahre gerecht zu werden. Besondere Berücksichtigung fanden dabei Fortschritte in den Bereichen Künstliche Intelligenz, maschinelles Lernen, Datenanalyse, Automatisierung und digitale Ethik.

Das Informatik-Curriculum orientiert sich an den Kompetenzbereichen der KMK-Strategie „Bildung in der digitalen Welt“ und legt besonderen Wert auf analytisches Denken, algorithmisches Verständnis, Problemlösekompetenz sowie kollaboratives Arbeiten in digitalen Projekten. Damit werden die Schüler*innen nicht nur auf digitalisierte Berufswelt bzw. auf ein technikbezogenes Studium vorbereitet, sondern auch auf eine verantwortungsbewusste Rolle in einer zunehmend digitalisierten Gesellschaft.

Inhaltsverzeichnis

1. Informatikunterricht in der Oberstufe im Zeitalter von KI und Digitalisierung.....	4
2. Strukturierung des Lehrplans	5
3. Kompetenzen im zeitgemäßen Informatikunterricht.....	6
4. Fachlehrplan der Jahrgangsstufen 11 bis 12	7
5. Bewertungskriterien und Überprüfbarkeit von Lernleistungen	15

1. Informatikunterricht in der Oberstufe im Zeitalter von KI und Digitalisierung

Digitale Technologien und Informationssysteme durchdringen heute nahezu alle Lebensbereiche. In der Oberstufe stellt die Auseinandersetzung mit Digitalisierung, globaler Vernetzung, Automatisierung und Künstlicher Intelligenz daher eine zentrale Bildungsaufgabe dar. Diese Entwicklungen bieten enorme Potenziale, werfen aber zugleich komplexe ethische, gesellschaftliche und sicherheitsrelevante Fragestellungen auf.

Informatik gilt in diesem Kontext nicht mehr als Zusatzqualifikation, sondern als unverzichtbare Schlüsselkompetenz, eine grundlegende Voraussetzung für akademischen Erfolg, berufliche Anschlussfähigkeit und gesellschaftliche Teilhabe. Der Informatikunterricht in der Sekundarstufe II vermittelt weit mehr als technisches Wissen: Er schult algorithmisches Denken, strukturiertes Problemlösen, Kreativität und kritische Reflexion.

Die Beschäftigung mit aktuellen Technologien (etwa maschinellem Lernen, algorithmischer Entscheidungsfindung oder KI-Systemen) erfolgt stets im Spannungsfeld von Funktionalität, Verantwortung und Wirkung. So erwerben die Schüler*innen ein tiefgreifendes Verständnis sowohl der technischen Grundlagen als auch der gesellschaftlichen Implikationen digitaler Systeme.

2. Strukturierung des Lehrplans

Der Informatikunterricht lebt von offenen Arbeitsformen und methodischer Vielfalt. Um auf die unterschiedlichen Lernvoraussetzungen einzugehen, sind projektorientierte Phasen fest im Curriculum verankert. Die Phasen 1 und 2 sind im Informatik-Curriculum der Sekundarstufe I integriert. In der Oberstufe fahren die Schüler*innen mit der dritten Phase fort. Zur besseren Orientierung werden folgende alle drei Phasen angeführt.

Phase 1: Digitale Grundlagen und Medienkompetenz (Klasse 5 – 7)

In dieser ersten Phase knüpft der Informatikunterricht an die Vorerfahrungen der Schülerinnen und Schüler an, die bereits in der Grundschule mit dem Tablet gearbeitet haben. Diese Kompetenzen werden weiterentwickelt: Sie lernen, digitale Geräte sicher zu nutzen, Informationen gezielt zu recherchieren und sich selbstständig in schulischen Lernumgebungen zurechtzufinden. Dabei stehen Themen wie Dateiverwaltung, Datenschutz und digitale Kommunikation im Mittelpunkt. Erste Erfahrungen im Programmieren, zum Beispiel mit blockbasierten Sprachen, fördern kreatives und strukturiertes Denken.

2. Phase: Programmieren, Vernetzen, Verstehen (Klasse 8 – 10)

In der Mittelstufe steht das kreative und strukturierte Lösen von Problemen im Mittelpunkt. Die Schülerinnen und Schüler entwickeln eigene Programme, arbeiten mit Robotik-Systemen und setzen sich mit objektorientierter Modellierung, Datenverarbeitung und Datenbanken auseinander. Dabei lernen sie sowohl technische als auch gesellschaftliche Aspekte digitaler Systeme kennen, von der Entwicklung intelligenter Anwendungen bis hin zur kritischen Reflexion über Datenschutz, Nachhaltigkeit oder algorithmische Entscheidungsprozesse.

Phase 3: Digitale Systeme entwickeln, Zukunft mitgestalten (Klasse 11 – 12)

In der Oberstufe wird Informatik zur analytischen und methodischen Disziplin: Die Lernenden vertiefen ihr Verständnis für komplexe Algorithmen, rekursive Strukturen, Netzwerke und maschinennahe Programmierung. Formale Sprachen, Softwareentwicklung, künstliche Intelligenz und ethische Fragestellungen stehen ebenso im Fokus wie praktische Projekte mit realitätsnahen Anwendungsszenarien. Ziel ist es, ein ganzheitliches Bild der digitalen Welt zu vermitteln.

Das wöchentlich Unterrichtsausmaß des Wahlpflichtfaches Informatik beträgt zwei Einheiten in der Oberstufe (11. und 12. Schulstufe).

3. Kompetenzen im zeitgemäßen Informatikunterricht

Zielsetzung und Bildungsauftrag

Informatik vermittelt Schülerinnen und Schülern zentrale Zukunftskompetenzen für eine digitale Welt. Sie lernen, mit Daten, Algorithmen und digitalen Systemen verantwortungsvoll umzugehen, Prozesse zu analysieren, Probleme modellbasiert zu lösen und technologische Entwicklungen wie Künstliche Intelligenz zu verstehen und kritisch zu reflektieren. Dabei fördert das Fach neben technischem Verständnis auch Kreativität, Teamarbeit, Urteilsfähigkeit und gesellschaftliche Verantwortung.

Der Unterricht ist projektorientiert und schülerzentriert aufgebaut. Unterschiedliche Lernniveaus werden durch gezielte Differenzierung berücksichtigt. Leistungsstarke Schülerinnen und Schüler können zusätzliche Aufgaben oder Mini-Projekte übernehmen, um gezielt Kompetenzen im Bereich der Softwareentwicklung und digitalen Systemgestaltung zu vertiefen. Auf diese Weise wird eine solide Grundlage geschaffen, um spätere Inhalte der Qualifikationsphase fachlich sicher zu bewältigen.

Zentrale Kompetenzbereiche

- **Analysieren und Modellieren:** Systeme und Problemstellungen strukturieren, abstrahieren und durch geeignete Modelle darstellen.
- **Implementieren und Anwenden:** Lösungen mit passenden Programmiersprachen und Werkzeugen umsetzen, testen und weiterentwickeln.
- **Bewerten und Vernetzen:** Technologische, gesellschaftliche und ethische Auswirkungen einschätzen und informatische Konzepte kontextübergreifend anwenden.
- **Kooperieren und Kommunizieren:** Lösungen im Team entwickeln, fachlich dokumentieren und präsentieren.

Gegenstandsbereiche

Grundlegende Themen wie Algorithmen, Datenstrukturen, Systemarchitekturen, Informationsdarstellung, Modellierung und Softwareentwicklung bilden das fachliche Fundament. Ergänzt werden sie durch gesellschaftlich relevante Fragestellungen wie Datenschutz oder ethische Aspekte Künstlicher Intelligenz. Im Mittelpunkt stehen praxisnahe Anwendungen, projektorientiertes Arbeiten und ein bewusster, reflektierter Umgang mit digitalen Technologien.

4. Fachlehrplan der Jahrgangsstufen 11 bis 12

Der Informatik-Lehrplan der Deutschen Schule Madrid ist spirallcurricular aufgebaut und orientiert sich an einer systematischen, altersgerechten Kompetenzentwicklung. Er begleitet die Schüler*innen kontinuierlich von der 5. Schulstufe bis zum Beginn der Oberstufe und baut schrittweise technische, methodische und gesellschaftlich relevante Fähigkeiten im digitalen Kontext auf.

In der Oberstufe sind die Informatik-Inhalte modular aufgebaut. Die Lernbereiche sind so angelegt, dass sie flexibel eingesetzt werden können, insbesondere, wenn aufgrund geringer Schülerzahlen ein jahrgangsübergreifender Unterricht in Klasse 11 und 12 notwendig wird.

Im Informatik-Schulcurriculum sind alle sechs Kompetenzbereiche des KMK-Rahmens „Kompetenzen in der digitalen Welt“¹ berücksichtigt. Welche konkreten Teilkompetenzen in welcher Klassenstufe behandelt werden, ist jeweils in der Spalte „KMK“ angegeben. Die in den Kompetenzformulierungen verwendeten Operatoren entsprechen in ihrer Funktion den Operatoren der KMK und lassen sich den drei Anforderungsbereichen (Reproduktion, Reorganisation/Transfer, Problemlösen/Bewerten) klar zuordnen. Sie ermöglichen eine differenzierte Leistungsbewertung auf allen Anspruchsniveaus. Für das Fach Informatik können dabei vereinzelt spezifische Operatoren verwendet werden, die nicht eins zu eins den Formulierungen der KMK entsprechen, was den fachlichen Besonderheiten geschuldet ist.

Hinweis: Im folgenden Informatik-Curriculum sind projektorientierte Arbeiten einzelnen Lernbereichen zugeordnet, um eine sinnvolle inhaltliche Verankerung und Orientierung zu bieten. Diese Zuordnung stellt jedoch eine Empfehlung dar und ist nicht verpflichtend: Lehrkräfte haben die pädagogische Freiheit, Projekte auch mit anderen Lernbereichen zu verknüpfen, sofern dies dem Lernprozess und den Zielkompetenzen dient.

¹ KMK Kultusminister Konferenz (2017): „Bildung in der digitalen Welt. Strategie der Kultusministerkonferenz. Online verfügbar unter: <https://www.kmk.org/themen/bildung-in-der-digitalen-welt/strategie-bildung-in-der-digitalen-welt.html> (abgerufen am 03.06.2025).

Informatik in den Jahrgangsstufen 11 & 12

Komplexe Informatiksystem: Architektur, Kommunikation, Intelligenz

Die Informatik in der Oberstufe vermittelt ein vertieftes Verständnis zentraler Konzepte, Methoden und Werkzeuge der digitalen Welt. Ein Schwerpunkt liegt auf der Entwicklung und Analyse von Algorithmen und Datenstrukturen, insbesondere rekursiven Verfahren, Listen, Bäumen und objektorientierten Ansätzen, unterstützt durch moderne KI-Tools zur Codegenerierung oder Visualisierung.

Im Bereich Rechnerarchitektur geht es um das Innenleben von Rechnersystemen, Zahlendarstellungen, Assemblerprogrammierung (Mikrocontroller). Grundlagen wie boolesche Algebra helfen beim Verständnis maschinennaher Prozesse.

Formale Sprachen, Automatenmodelle und Konzepte der Berechenbarkeit schaffen das Fundament für die theoretische Informatik. Dabei werden auch die Grenzen algorithmischer Lösbarkeit und Aspekte der Laufzeitanalyse behandelt.

Der Lehrplan der Jahrgangsstufen 11 und 12 ist modular aufgebaut. Die sechs Lernbereiche sind so angelegt, dass sie flexibel eingesetzt werden können – insbesondere, wenn aufgrund geringer Schülerzahlen ein jahrgangsübergreifender Unterricht in Klasse 11 und 12 notwendig wird. Dabei ist sichergestellt, dass die Lernbereiche in beliebiger Reihenfolge behandelt werden können, da sie inhaltlich nicht zwingermaßen aufeinander aufbauen.

Lernbereich 1: Algorithmik mit rekursiven Konzepten und Datenstrukturen (20 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Rekursive Konzepte und Problemlösungen - rekursive und iterative Lösungsstrategien vergleichen (z. B. Fibonacci, Türme von Hanoi) - rekursive Methoden implementieren mit korrekter Abbruchbedingung und Aufrufstruktur - Nutzung von KI-Tools zur Generierung, Visualisierung oder Kommentierung rekursiver Algorithmen (z. B. mit GitHub Copilot, VibeCoding) - Analyse rekursiver Abläufe mit interaktiven Visualisierungen und Debugging-Assistenz	- analysieren und implementieren rekursive Strukturen zur Lösung klassischer algorithmischer Probleme und reflektieren deren Effizienz. - Verwenden KI-basierte Werkzeuge zur Codevervollständigung, zum Debugging, zur Visualisierung von Aufrufstrukturen oder zur Generierung von Beispielen – dabei reflektieren sie die Grenzen maschineller Unterstützung. - entwickeln, implementieren und bewerten dynamische Datenstrukturen wie Listen oder Bäume in objektorientierten Umgebungen. - modellieren komplexere Systeme mit Vererbung, Generalisierung und Polymorphie und übertragen diese auf reale Anwendungsszenarien. - bewerten Unterschiede zwischen rekursiven und iterativen Verfahren sowie zwischen statischen und dynamischen Datenstrukturen. - verwenden KI-Tools verantwortungsvoll und reflektiert zur Unterstützung von Problemlösungs- und Entwicklungsprozessen. - dokumentieren und hinterfragen maschinell generierte Lösungen, verbessern diese manuell und begründen eigene Implementierungsentscheidungen nachvollziehbar.	5.5.3 5.1.3 5.2.1 6.2.2 5.5.2 5.5.3 5.5.2 5.5.3 5.5.2 6.2.2 6.2.2 4.2.1 3.2.2 5.1.3
Rekursive Datenstrukturen - einfach verkettete Listen als dynamische Struktur (Einfügen, Entfernen, Suchen, Traversieren) - Anwendung des Kompositum-Musters zur Trennung von Struktur und Daten - Modellierung typischer Beispiele mit Hilfe von KI-gestützten Systemen zur Codevervollständigung - Simulation von Listenstrukturen und Stapeln in virtuellen Sandbox-Umgebungen (z. B. VibeCoding) - Diskussion: Wann verstehen – wann automatisieren?		
Bäume als strukturierte Datenmodelle - Aufbau von Binärbäumen: Knoten, Kanten, Höhe, Tiefe, Blätter - Traversierungsverfahren: Preorder, Inorder, Postorder - rekursive Algorithmen zum Einfügen, Suchen und Durchlaufen - Einsatz von KI-Assistenten zur Generierung von Testdaten, Visualisierung von Baumstrukturen oder zur Vereinfachung rekursiver Traversierungslogik - kritische Bewertung generierter Lösungen durch menschliche Analyse		
Objektorientierte Modellierung mit Generalisierung - Vererbung, Spezialisierung und Generalisierung grafisch und im Code modellieren - abstrakte Klassen und Methoden einsetzen - Polymorphie zur flexiblen Verhaltensdefinition verwenden - Wiederverwendbarkeit durch KI-gestützte Vorschläge von Designmustern oder Refactoring-Ideen fördern - Unterstützung durch intelligente IDEs (z. B. JetBrains AI, Copilot) bei Methodensignaturen, Abstraktion, Kapselung		

Lernbereich 2: Netzwerkkommunikation (15 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Kommunikation in Netzwerken und im Internet - Struktur und Funktionsweise des Internets: Client, Server, Router - Adressierung von Geräten im Netzwerk (MAC-Adresse, IP-Adresse, Port, DNS) - Schichtenmodell: Anwendungen aufschlüsseln in logische Ebenen - Protokolle als Regeln der Kommunikation (z. B. HTTP, FTP, TCP, DNS) - Webanwendungen als Anwendungsfall: Zusammenspiel von Client & Server (z. B. Fahrplanauskunft, Login-Formular) - Einsatz moderner Web-Technologien (z. B. WebSockets, REST-APIs, JSON) - Umgang mit persönlichen Daten: Cookies, Tracking, DSGVO, Privacy-by-Design	- beschreiben die Architektur des Internets und erklären grundlegende Prinzipien der Datenübertragung, einschließlich Adressierung und Protokolle. - analysieren Kommunikationsvorgänge mithilfe des Schichtenmodells und erkennen die Bedeutung standardisierter Protokolle für reibungslosen Datenaustausch. - erläutern das Zusammenspiel von Client und Server anhand praktischer Beispiele und entwickeln ein Verständnis für dynamische Webanwendungen. - beurteilen Risiken und Schutzmöglichkeiten im Zusammenhang mit persönlicher Datennutzung (z. B. bei Webdiensten, Cookies, Tracking) und reflektieren die gesellschaftliche Bedeutung digitaler Selbstbestimmung. - analysieren Prozesse in digitalen Systemen, strukturieren diese in logisch getrennte Abläufe und erkennen, wie Kommunikation und Koordination zwischen parallelen Prozessen organisiert sein müssen. - modellieren nebenläufige Abläufe mithilfe von Petrinetzen, analysieren diese auf mögliche Konflikte oder Verklemmungen und entwickeln geeignete Lösungen. - übertragen ihre Kenntnisse auf reale Anwendungssituationen – z. B. in der Lagerlogistik, Verkehrssteuerung oder bei der Nutzung gemeinsamer digitaler Ressourcen in Cloud-Diensten.	5.5.1 5.5.1 5.5.1 5.5.3 4.2.1 6.2.6 5.5.3 5.5.3 5.1.3 5.5.2 6.2.2
Analyse und Modellierung von Kommunikationsvorgängen - Kommunikation zwischen Prozessen als Basis moderner Systeme (z. B. Microservices, verteilte Anwendungen, IoT) - Fehlerursachen in Netzwerken identifizieren (z. B. durch falsche IP-Zuweisung oder Portblockierung) - Erkennen der Bedeutung von Protokollen zur Prozesssteuerung in Systemarchitekturen		
Nebenläufigkeit und Prozesssynchronisation - Begriffe: Prozess, Nebenläufigkeit, Synchronisation, Verklemmung - Darstellung und Analyse nebenläufiger Systeme mithilfe von Petrinetzen (Stelle, Transition, Marke, Kapazität) - Anwendung: Modellierung von Geschäftsabläufen, Verkehrssteuerung, Robotersteuerung - Klassische Probleme: Erzeuger-Verbraucher-Problem, Leser-Schreiber-Problem - Reflektion: Bedeutung synchronisierter Abläufe in modernen Cloud- und Serverumgebungen		

Lernbereich 3: Funktionsweise eines Rechners (25 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Rechnerarchitektur und Steuerung - Von-Neumann-Architektur: Steuerwerk, Rechenwerk, Speicher, Busse - Mikrocontrollerarchitektur im Vergleich (z. B. Harvard-Architektur) - Steuerungseinheiten, Datenpfade, Maschinenbefehle - Grundlagen der Datenübertragung: Adressbus, Datenbus, Steuerbus	- beschreiben den Aufbau und die Funktionsweise moderner Rechnersysteme auf Basis der Von-Neumann- und Harvard-Architektur und vergleichen deren Einsatzbereiche. - wandeln Zahlen sicher zwischen dezimalem, binärem und hexadezimalen System um und nutzen diese zur Darstellung, Berechnung und Steuerung in maschinennahen Anwendungen.	5.5.1
Zahlendarstellung und logische Operationen - Zahlensysteme: Binär, Dezimal, Hexadezimal, Umwandlung und Anwendung - Boolesche Algebra: UND, ODER, NICHT; Wahrheitstabellen - Anwendung in Steuerungs- und Vergleichsoperationen - Maschinennahe Programmierung - Register, Speicher, Peripherie, Aufbau eines Mikrocontrollers - Assemblerprogrammierung: Registermanipulation, Schleifen, Verzweigungen - Anwendungen: LED-Steuerung, Tastersteuerung, einfache Zähler	- analysieren und implementieren einfache Steuerungsaufgaben mithilfe von Assemblercode auf Mikrocontroller-Basis und testen deren Funktionalität mit realer oder simulierter Hardware. - Modellieren logische Abläufe mithilfe boolescher Algebra und Wahrheitstabellen und überführen sie in digitale Steuerungslogik - übertragen strukturiert formulierte Algorithmen in Assemblersprache und reflektieren deren Ablauf, Steuerung und Effizienz auf Maschinenebene.	5.5.3 5.1.3 5.5.2 5.5.3 6.2.2
Vom Algorithmus zur Umsetzung in Assembler - Übersetzung strukturierter Programme in Assemblersprache - Steuerstrukturen (IF, WHILE ...) in Maschinenbefehle - Taktzyklen, Ablaufsteuerung, Maschinenebene - Reflexion zur Effizienz und Abstraktionsebene maschinennaher Programmierung		

Lernbereich 4: Künstliche Intelligenz (15 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Vertiefung neuronaler Netze und Lernprozesse - Vergleich einfacher (Perzeptron) und komplexer Netzarchitekturen (z. B. Multi-Layer-Perzeptron, CNN, RNN) - Detaillierte Analyse der Vorwärtsausbreitung und Backpropagation mit Gewichtsanzpassung - Bedeutung von Lernrate, Aktivierungsfunktionen, Optimierungsverfahren (z. B. Gradient Descent) - Erweiterte Simulationen in KI-Entwicklungsumgebungen (z. B. TensorFlow Playground, VibeCoding Advanced) Komplexe Lernverfahren und Modellbewertung - Vergleich überwachter, unüberwachter und bestärkender Lernverfahren anhand realitätsnaher Szenarien - Vertiefung: Clustering, Entscheidungsbäume, k-Nächste-Nachbarn, Reinforcement Learning (z. B. in spielerischen Agentensystemen) - Einsatz von Metriken zur Modellbewertung (Genauigkeit, Precision, Recall, F1-Score) - Auseinandersetzung mit Overfitting, Underfitting, Regularisierung, Cross-Validation Anwendung, Reflexion und Verantwortung - Analyse konkreter KI-Systeme im Alltag (z. B. Spracherkennung, Empfehlungssysteme, Scoring-Modelle) - Erkennen und Diskutieren systemischer Risiken: algorithmische Voreingenommenheit, Datenqualität, Skalierungsrisiken - Vertiefte ethische Reflexion: Entscheidungsautomatisierung, Verantwortung, Nachvollziehbarkeit, KI-Governance - Gesellschaftliche Herausforderungen: Deepfakes, KI-generierte Inhalte, Automatisierung von Bildung/Arbeit	- analysieren den Aufbau und die Funktionsweise verschiedener neuronaler Netztypen und simulieren deren Lernverhalten mit modernen Tools. - reflektieren und erklären die Rolle von Parametern und Trainingsdaten für die Leistungsfähigkeit von Modellen. - vergleichen und bewerten maschinelle Lernverfahren auf Basis mathematischer Metriken und begründen die Eignung für verschiedene Anwendungsfelder. - implementieren einfache KI-Anwendungen selbstständig in schulgeeigneten Entwicklungsumgebungen - diskutieren Risiken und Grenzen des KI-Einsatzes, insbesondere in Bezug auf Diskriminierung, Transparenz und gesellschaftliche Verantwortung. - formulieren ethische Positionen zu KI-gestützten Entscheidungen und entwickeln konstruktive Lösungsansätze für einen verantwortungsvollen Umgang mit KI-Systemen.	5.5.1 5.2.1 6.2.2 1.2.1 6.1.3 5.5.3 5.2.1 6.2.6 4.2.1 6.2.6 6.2.5

Lernbereich 5: Formale Sprachen, Automaten (10 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Formale Sprachen und Grammatiken - Formale Sprachen als Mengen von Zeichenketten (Alphabet, Wort, Syntax, Semantik) - Definition regulärer Sprachen und formaler Grammatiken - Notation mit EBNF und Syntaxdiagrammen - Ableitung von Wörtern durch Produktionsregeln und Ableitungsbäume Endliche Automaten - Aufbau und Funktionsweise deterministischer und nichtdeterministischer endlicher Automaten - Zustandsmengen, Übergangsfunktionen, Start- und Endzustände, Fehlerzustand - Implementierung einfacher Automaten zur Worterkennung regulärer Sprachen - Grenzen regulärer Sprachen (z. B. bei verschachtelten Strukturen) Komplexität und Laufzeitanalyse - Aufwandsschätzung für Algorithmen: Laufzeitklassen (linear, quadratisch, exponentiell etc.) - Fallunterscheidungen: Best Case, Worst Case, Average Case - Verfahren zur Laufzeitanalyse: Zählverfahren, Zeitmessung - Sicherheitsaspekte bei ineffizienten Verfahren (z. B. Brute-Force-Angriffe) Nicht berechenbare Probleme - Einführung in das Halteproblem - Widerspruchsbeweis zur Unentscheidbarkeit - Bedeutung für Theorie und Praxis: Grenzen automatisierter Verfahren Verbindung zur Informatikpraxis - Erkennen formaler Sprachen im Alltag (z. B. Syntax von Programmiersprachen, E-Mail-Adressen) - Relevanz formaler Methoden für Compilerbau, Spracherkennung und Verifikation - Reflexion der Bedeutung von Komplexitäts- und Berechenbarkeitsgrenzen für die Softwareentwicklung	- analysieren und formulieren Regeln für formale Sprachen und setzen diese mithilfe von EBNF und Syntaxdiagrammen um - entwerfen und implementieren endliche Automaten zur Erkennung regulärer Sprachen und beurteilen deren Grenzen. - vergleichen Sprachen hinsichtlich ihrer Ausdruckskraft und zeigen an Beispielen, welche Sprachen nicht regulär sind - untersuchen den Laufzeitaufwand von Algorithmen durch Zählverfahren und Zeitmessung und ordnen diese in Komplexitätsklassen ein - erklären am Beispiel des Halteproblems die Grenzen automatisierter Problemlösbarkeit - bewerten die sicherheitsrelevanten Konsequenzen ineffizienter Algorithmen und reflektieren die Bedeutung theoretischer Grundlagen für die Praxis	5.5.2 5.5.3 6.2.2 5.5.1 5.5.3 5.5.1 6.2.2 4.1.1

Lernbereich 6: Softwareentwicklung (35 UE)

Inhalt	Kompetenzen: Die Schüler*innen ...	KMK
Projektorientiertes Arbeiten mit modernen Entwicklungsmethoden - Vergleich traditioneller Modelle (z. B. Wasserfall) mit agilen Ansätzen (Scrum, Kanban) - Praktische Umsetzung agiler Methoden: User Stories, Sprints, Stand-Ups, Retrospektiven - Projektplanung mit Meilensteinen, Zeitplanung und Rollenzuteilung	- planen, organisieren und realisieren ein Softwareprojekt im Team unter Verwendung agiler Entwicklungsmethoden und setzen dabei moderne Werkzeuge ein - wenden KI-gestützte Tools zur Ideenfindung, Codegenerierung oder Testunterstützung an und bewerten deren Nutzen und Grenzen kritisch - entwickeln grafische Benutzeroberflächen und binden einfache KI-Komponenten oder Online-Dienste in ihre Softwareprodukte ein - dokumentieren Projektverlauf, Entscheidungen und Quellcode in nachvollziehbarer und kollaborativer Weise - testen und verbessern Software iterativ durch Refactoring und verschiedene Testverfahren und reflektieren Qualität und Wartbarkeit ihrer Lösung - reflektieren über Herausforderungen kooperativer Entwicklung (z. B. Codezusammenführung, Konfliktmanagement) und zeigen Verantwortung im Umgang mit externen Ressourcen (Open-Source, Lizenzen, Datenquellen) - analysieren die Rolle von KI in der Softwareentwicklung aus technischer und ethischer Perspektive und entwickeln ein Bewusstsein für Mensch-in-der-Schleife-Prinzipien	2.3.1 3.1.2 5.1.3 5.2.1 6.2.2 5.5.3 3.1.2 3.2.2 2.3.2 5.1.3 2.4.1 3.3.2 6.2.6 6.2.5 4.2.1
Softwareentwicklung im Team – kooperativ und versionssicher - Nutzung von Versionskontrollsystemen (z. B. GitHub, GitLab) - Kollaboration über Cloud-basierte Entwicklungsplattformen (z. B. Replit, Codespaces) - Dokumentation mit Markdown, Readme-Dateien und Wikis		
Entwicklung grafischer Anwendungen und KI-gestützter Systeme - Gestaltung und Programmierung moderner Benutzeroberflächen (z. B. Web-UI mit HTML/CSS/JS oder GUI-Frameworks) - Einführung in Low-Code-/No-Code-Tools zur schnellen UI-Erstellung - Integration einfacher KI-Komponenten		
Architektur, Wartung und Test moderner Anwendungen - Architekturkonzepte: Model-View-Controller, Microservices (in Grundzügen) - Refactoring und Clean Code: Verständlichkeit, Erweiterbarkeit, Wartbarkeit - Testverfahren: manuell, automatisch, testgetriebene Entwicklung		
Einsatz von KI in der Entwicklung - Nutzung von KI-Coding-Assistenten zur Unterstützung bei der Codierung - Bewertung von KI-generiertem Code: Chancen, Risiken und Verantwortung - Automatisierte Tests durch KI, Vorschlags- und Fehleranalysefunktionen		

5. Bewertungskriterien und Überprüfbarkeit von Lernleistungen

Die Fachkonferenz legt folgende Grundsätze für die Bewertung im Fach Informatik fest:

- Verhältnis von schriftlichen und mündlichen Leistungen:
Es wird festgelegt, wie schriftliche (z. B. Klassenarbeiten) und mündliche Leistungen (z. B. Beiträge im Unterricht, Präsentationen) gewichtet werden sollen, um die Gesamtnote zu ermitteln.
- Transparente Bewertung von Klassenarbeiten:
Es werden prozentuale Vorgaben oder konkrete Punktabgaben für die Notenvergabe in Klassenarbeiten gemacht.
- Weitere Aspekte der Bewertung:
Es wird vereinbart, wie sprachliche Ausdrucksfähigkeit in die Bewertung einfließt oder in welchem Umfang grundlegendes Fachwissen abgefragt werden soll.

5.1. Definition der Anforderungsbereiche (gem. Einheitlicher Prüfungsforderungen in der Abiturprüfung):

- Anforderungsbereich I: Reproduktion
Wiedergabe einfacher Fachinhalte, Anwendung grundlegender Methoden, Darstellung in vorgegebener Form
- Anforderungsbereich II: Reorganisation und Transfer
Umstrukturierung und Anwendung komplexerer Inhalte und Methoden, situationsgerechte Kommunikation, einfache Bezüge und Bewertungen.
- Anforderungsbereich III: Problemlösen und Bewerten
Selbstständige Anwendung komplexer Inhalte und Methoden, Auswahl passender Darstellungsformen, begründete Bewertungen und Verknüpfungen von Sachverhalten

5.2. Leistungsbewertung

Jahrgang	Art der Leistungsbewertung	Verhältnis schriftlich : mündlich
11	• Schriftliche Leistungen: mindestens eine Klausur pro Halbjahr, bei projektorientierten Vorhaben kann eine fachlich gleichwertige Projektleistung die Klausur ersetzen, sofern sie dokumentiert, reflektiert und bewertet wird. • Mündliche Leistungen: Unterrichtsbeiträge, Präsentationen, Kurztests, praktische Arbeiten am Rechner, ggf. Portfolioelemente.	1:1
12	• Schriftliche Leistungen: in jedem Kurshalbjahr eine Klausur; Projektbasierte Leistungen sind möglich, sofern sie im Umfang und Anspruch einer Klausur entsprechen. • Mündliche Leistungen: Unterrichtsbeiträge, Präsentationen, Kurztests, praktische Arbeiten am Rechner, ggf. fachliche Reflexionen oder Protokolle.	1:1