LEHRVERANSTALTUNGSBESCHREIBUNG

Formale Sprachen und Kompiliertechniken

Akademisches Jahr 2025-2026

1. Angaben zum Programm

1.1. Hochschuleinrichtung	**Universitatea Babes-Bolyai**
1.2. Fakultät	**Mathematik und Informatik**
1.3. Department	**Informatik**
1.4. Fachgebiet	**Informatik**
1.5. Studienform	**Bachelor**
1.6. Studiengang / Qualifikation	**Informatik in deutscher Sprache**
1.7. Form des Studiums	**Präsenzstudium**

2. Angaben zum Studienfach

2.1. LV-Bezeichnung	**Formale Sprachen und Kompiliertechniken**					Code der LV	MLG5023
2.2. Lehrverantwortlicher – Vorlesung	Prof. dr. Klaus Dohmen						
2.3. Lehrverantwortlicher – Seminar	Prof. dr. Klaus Dohmen						
2.4. Studienjahr	3	2.5. Semester	1	2.6. Prüfungsform	E	2.7. Art der LV	Pflichtfach

3. Geschätzter Workload in Stunden

3.1. SWS	**4**	von denen: 3.2 Vorlesung	**2**	3.3. Seminar/Übung/Projekt	**2+2**
3.4. Gesamte Stundenanzahl im Lehrplan	**84**	von denen: 3.5 Vorlesung	**28**	3.6 Seminar/Übung/Projekt	**56**
Verteilung der Studienzeit:					**Std.**
Studium nach Handbücher, Kursbuch, Bibliographie und Mitschriften					10
Zusätzliche Vorbereitung in der Bibliothek, auf elektronischen Fachplattformen und durch Feldforschung					10
Vorbereitung von Seminaren/Übungen, Präsentationen, Referate, Portfolios und Essays					10
Tutoriat					8
Prüfungen					3
Andere Tätigkeiten:					
3.7. Gesamtstundenanzahl Selbststudium	**41**				
3.8. Gesamtstundenanzahl / Semester	**150**				
3.9. Anrechnungspunkte	**5**				

4. Voraussetzungen (falls zutreffend)

4.1. zur Lehrveranstaltung	Datenstrukturen und Algorithmen
4.2. kompetenzbezogene	Programmingskills

5. Bedingungen (falls zutreffend)

5.1. zur Durchführung der Vorlesung	Vorlesungsraum, Beamer, Laptop
5.2. zur Durchführung des Seminars / der Übung	Computerraum

6.1. Spezifische erworbene Kompetenzen[1]

Berufliche/Wesentliche Kompetenzen	K 4.1 Definieren der Grundkonzepte und Prinzipien der Informatik, sowie der mathematischen Theorien und Modelle K 4.2 Interpretation der formalen Modelle der Mathematik und Informatik K 4.3 Identifizierung der geeigneten Modelle und Methoden für die Lösung realer Probleme K 4.4 Anwendung der Simulierungen für die Untersuchung der Verhaltensweise der angewandten Modelle und Bewertung der Ergebnisse K4.5 Einbauen der formalen Modelle in geeignete Anwendungen der spezifischen Gebiete
Transversale Kompetenzen	TK1 Anwendung der Regeln für gut organisierte und effiziente Arbeit, für verantwortungsvolle Einstellungen gegenüber der Didaktik und der Wissenschaft, für kreative Förderung des eigenen Potentials, mit Rücksicht auf die Prinzipien und Normen der professionellen Ethik TK3 Anwendung von effizienten Methoden und Techniken für Lernen, Informieren und Recherchieren, für das Entwicklen der Kapazitäten der praktischen Umsetzung der Kenntnisse, der Anpassung an die Bedürfnisse einer dynamischen Gesellschaft, der Kommunikation in rumänischer Sprache und in einer internationalen Verkehrssprache

6.2. Lernergebnisse

Kenntnisse	Der/Die Absolvent/in verfügt über Kenntnisse in den Bereichen Programmierung, Mathematik, Ingenieurwesen und Technologie und besitzt die Fähigkeiten, diese zur Erstellung komplexer Informationstechnologiesysteme zu nutzen. Der/Die Absolvent/in verfügt über das Wissen, geeignete instruktionale Verfahren (Lehrmethoden) auszuwählen und anzuwenden, um den Prozess der Wissensaneignung zu erleichtern.
Fähigkeiten	Der/Die Absolvent/in ist in der Lage, Methoden, Algorithmen, Paradigmen und Techniken aus verschiedenen Bereichen der Informatik zu präsentieren und zu erklären. Der/Die Absolvent/in ist in der Lage, komplexe Probleme zu identifizieren und damit verbundene Sachverhalte zu untersuchen, um Lösungsoptionen zu entwickeln und Lösungen zu implementieren. Der/Die Absolvent/in ist in der Lage, vielfältige Informationen zu kombinieren, um Lösungen zu formulieren und Ideen für die Entwicklung neuer Produkte und Anwendungen zu generieren..
Verantwortung und Autonomie	Der/Die Absolvent/in besitzt die Fähigkeit, allgemeine Regeln auf spezifische Probleme anzuwenden und relevante Lösungen zu erarbeiten. Der/Die Absolvent/in besitzt die Fähigkeit, Programmierparadigmen (prozedural, objektorientiert, funktional) auszuwählen und anzuwenden, um Softwareanwendungen zu entwickeln, die für den spezifischen Bereich der zu entwickelnden Anwendung geeignet sind.

7. Ziele (entsprechend der erworbenen Kompetenzen)

7.1 Allgemeine Ziele der Lehrveranstaltung	Das Erlernen und Verstehen wie man Compiler aufbaut. Verbesserung der Programmierfähigkeiten.
7.2 Spezifische Ziele der Lehrveranstaltung	Kenntnisse über ein compiler back-end. Aneignen der grundlegenden Begriffe der formalen Sprachen. Aneignen der grundlegenden Begriffe über Compiler.

8. Inhalt

8.1 Vorlesung	Lehr-und Lernmethode	Anmerkungen

[1] Man kann Kompetenzen oder Lernergebnisse, oder beides wählen. Wenn nur eine Option ausgewählt wird, wird die Tabelle für die andere Option gelöscht, und die beibehaltene Option erhält die Nummer 6.

1. Einführung in die Compiler Implementierung. Grammatiken und Sprachen.	Vortrag, Erklärung, Debatte, praktische Beispiele	
2. Lektische Analysis.	Vortrag, Erklärung, Debatte, praktische Beispiele	
3. Reguläre Grammatiken, endliche Automata.	Vortrag, Erklärung, Debatte, praktische Beispiele	
4. Kontext unabhängige Grammatiken.	Vortrag, Erklärung, Debatte, praktische Beispiele	
5. Push-down Automata.	Vortrag, Erklärung, Debatte, praktische Beispiele	
6. Spezielle Grammatiken – LL(k) Grammatiken.	Vortrag, Erklärung, Debatte, praktische Beispiele	
7. Spezielle Grammatiken – LR(k) Grammatiken.	Vortrag, Erklärung, Debatte, praktische Beispiele	
8. Spezielle Grammatiken – LR(k) Grammatiken. SLR, LR(1) und LALR Analysis.	Vortrag, Erklärung, Debatte, praktische Beispiele	
9. Compiler Struktur.	Vortrag, Erklärung, Debatte, praktische Beispiele	
10. Lektischer Analysis Generator lex/flex.	Vortrag, Erklärung, Debatte, praktische Beispiele	
11. Beweise, Konstruktionen und Anwendungen. Äquivalenz der endlichen Automata mit den regulären Grammatiken.	Vortrag, Erklärung, Debatte, praktische Beispiele	
12. Beweise, Konstruktionen und Anwendungen. Eigenschaften regulärer Sprachen.	Vortrag, Erklärung, Debatte, praktische Beispiele	
13. Semantische Analysis.	Vortrag, Erklärung, Debatte, praktische Beispiele	
14. Anwendungen formaler Mechanismen in der semantischen Analysis.	Vortrag, Erklärung, Debatte, praktische Beispiele	

Literatur in deutscher Sprache

[1] C. WAGENKNECHT, HIELSCHER M., Formale Sprachen, abstrakte Automaten und Compiler, Vieweg Teubner, 2009.

[2] ASTEROTH, A., BAIER, C., Theoretische Informatik, eine Einführung in Berechnbarkeit, Komplexität und formale Sprachen, Pearson Studium, 2002.

[3] HROMKOVIC, J., Theoretische Informatik, Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie, Vieweg Teubner, 2011.

Literatur in englischer Sprache

[1] K.D. COOPER, L. TORCZON - Engineering a Compiler, Elsevier Science & Technology, 2011.

[2] A.V. AHO, D.J. ULLMAN - Principles of compiler design, Addison-Wesley, 1978.		
8.2 Übung	Lehr-und Lernmethode	Anmerkungen
1-2. Spezifizierung einer Programmiersprache. Die BNF Notation.	Beispiele, Diskussionen, Teamarbeit	
3-4. Grammatiken, die von Grammatiken erzeute Sprache, die Grammatik einer Sprache.	Beispiele, Diskussionen	
5-6. Endliche Automata	Beispiele, Diskussionen	
7-8. Kontext unabhängige Grammatiken. Syntaktische Analysis LL(1).	Beispiele, Diskussionen, Teamarbeit	
9-10. Syntaktische Analysis LR(k).	Beispiele, Diskussionen, Teamarbeit	
11-12. Spracheigenschaften. Beweise und Anwendungen.	Beispiele, Diskussionen, Teamarbeit	
13-14. APD. Grammatiken vom Typ 1, 2 und 3 (Chomsky Hierarchie).	Diskussionen	
Literatur in deutscher Sprache [1] C. WAGENKNECHT, HIELSCHER M., Formale Sprachen, abstrakte Automaten und Compiler, Vieweg Teubner, 2009. [2] ASTEROTH, A., BAIER, C., Theoretische Informatik, eine Einführung in Berechnbarkeit, Komplexität und formale Sprachen, Pearson Studium, 2002. [3] HROMKOVIC, J., Theoretische Informatik, Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie, Vieweg Teubner, 2011. Literatur in englischer Sprache [1] K.D. COOPER, L. TORCZON - Engineering a Compiler, Elsevier Science & Technology, 2011. [2] A.V. AHO, D.J. ULLMAN - Principles of compiler design, Addison-Wesley, 1978.		
8.3 Labor	Lehr-und Lernmethode	Anmerkungen
1. Spezifizieren einer mini Programmiersprache und Implementieren eines lexikalen Analysators. Spezifikation der mini Programmiersprache	Beispiele, Diskussionen, Teamarbeit	
2. Spezifizieren einer mini Programmiersprache und Implementieren eines lexikalen Analysators. Implementation der Hauptfunktionen des Analysators.	Beispiele, Diskussionen, Teamarbeit	
3. Spezifizieren einer mini Programmiersprache und Implementieren eines lexikalen Analysators. Organisation der Symbolentafel.	Beispiele, Diskussionen, Teamarbeit	
4. Spezifizieren einer mini Programmiersprache und Implementieren eines lexikalen Analysators.	Beispiele, Diskussionen, Teamarbeit	

Hauptprogramm. Testen und Abgabe.		
Endliche Automata: Verifikation der Akzeptierung einer Datensequenz. Das Wählen der Datenstrukturen und die Architektur der Anwendung.	Beispiele, Diskussionen, Teamarbeit	
Endliche Automata: Implementierung, Testen und Abgabe.	Beispiele, Diskussionen, Teamarbeit	
Endliche Automata: Anpassen des Programms für die lektische Analysis um die Benutzung endlicher Automata zu erlauben. Bestimmen der Sequenzen lektischer Atome.	Beispiele, Diskussionen, Teamarbeit	
Implementierung eines syntaktischen Analysators: Datenstruktur Auswahl und Programmarchitektur.	Beispiele, Diskussionen, Teamarbeit	
Implementierung eines syntaktischen Analysators: Hauptfunktionen.	Beispiele, Diskussionen, Teamarbeit	
Implementierung eines syntaktischen Analysators: Hauptprogramm und Modulintegration.	Beispiele, Diskussionen, Teamarbeit	
Implementierung eines syntaktischen Analysators: Testen und Fehlerbehebung.	Beispiele, Diskussionen, Teamarbeit	
Implementierung eines syntaktischen Analysators: Abgabe.	Beispiele, Diskussionen, Teamarbeit	
Anwendung von lex/flex + yac/bison: Implementierung.	Beispiele, Diskussionen, Teamarbeit	
Anwendung von lex/flex + yac/bison: Testen und Abgabe.	Beispiele, Diskussionen, Teamarbeit	
Literatur [1] K.D. COOPER, L. TORCZON - Engineering a Compiler,Elsevier Science & Technology, 2011. [2] Online-Materialien, zB. http://dinosaur.compilertools.net/		

9. Verbindung der Inhalte mit den Erwartungen der Wissensgemeinschaft, der Berufsverbände und der für den Fachbereich repräsentativen Arbeitgeber

Diese Vorlesung wird an international bekannten Universitäten im Fachgebiet Informatik.

Der Inhalt des Kurses gilt als wichtiger Teil der Programmierkenntnisse der Informatiker in Software-Unternehmen..

10. Prüfungsform

Veranstaltungsart	10.1 Evaluationskriterien	10.2 Evaluationsmethoden	10.3 Anteil an der Gesamtnote
10.4 Vorlesung	Grundkenntnisse.	Schriftliche Arbeit	75%
10.5 Seminar / Übung	Algorithmenanwendung	Labor Arbeiten	25%
10.6 Minimale Leistungsstandards			
Für das Bestehen der Prüfung muss die Mindestnote 5 erzielt werden (bei der schriftlichen Arbeit, bzw. beim Lösen der Laboraufgaben).			

11. SDD-Nachhaltigkeits-Logos (Sustainable Development Goals)[2]

Nicht anwendbar.

Ausgefüllt am:
17.04.2025

Vorlesungsverantwortlicher

Prof. dr. Klaus Dohmen

Seminarverantwortlicher

Prof. dr. Klaus Dohmen

Genehmigt im Department am:
...

Departmentleiter

Conf. dr. Adrian STERCA

[2] Bitte belassen Sie nur die Logos, die entsprechend den *Regularien zu Anwendung der Nachhaltigkeits-Logos im akademischen Betrieb* dem jeweiligen Studienfach entsprechen und löschen Sie diejenigen Logos, inklusive das allgemeine *Nachhaltigkeits-Logo* falls dieses nicht zutrifft. Falls keines der Logos für das Studienfach anwendbar ist, löschen Sie alle mit der Angabe „*nicht anwendbar*".