Programmieren und Software Engineering

Die Studierenden können

- Bereich Theoretische Informatik:
  - in verschiedenen Zahlensystemen Grundrechenoperationen ausführen, zwischen Zahlensystemen konvertieren, Fehler analysieren und diese programmtechnisch anwenden
  - Graphen in geeigneter Form darstellen sowie analysieren und Probleme graphentheoretisch modellieren sowie geeignete Strategien zu deren Lösung angeben und diese implementiere
  - Standardalgorithmen für eine konkrete Problemstellung auswählen.
  - umgangssprachliche Sätze in prädikatenlogische Formeln oder eine mehrwertige Logik übertragen und umgekehrt
  - formale Sprachen, Grammatiken und Syntaxanalyseverfahren anwenden
  - Algorithmen verstehen und diese in einer Programmiersprache umsetzen sowie für komplexe Aufgabenstellungen Algorithmen kombinieren und adaptieren.
- Bereich Softwareentwicklung und -design:
  - Zusammenhänge eines Problems erfassen und mit metasprachlichen Methoden darstellen
  - einfache Problemstellungen in Programme umsetzen
  - Informationen in vorgegebenen Datenstrukturen darstellen
  - die Effizienz unterschiedlicher Datenstrukturen bezüglich Datenumfang, Sicherheit und Aufwand beurteilen
  - einfache Datenstrukturen und Algorithmen implementieren
  - vorgegebene Vererbungshierarchien entwickeln und gemeinsam mit grundlegenden Klassen der Bibliotheken zu Lösungen von Aufgaben einsetzen
  - Informationen in vorgegebenen Datenstrukturen darstellen
  - in Programmen externe Datenzugriffe realisieren und mit anderen Programmen kommunizieren
  - die Effizienz unterschiedlicher Datenstrukturen bezüglich Datenumfang, Sicherheit und Konvertierungsaufwand beurteilen
  - vorgegebene Userinterfaces mit Hilfe fertiger Controls erstellen und auf Benutzereingaben angemessen reagieren
  - neue Userinterfaces für Client-Anwendungen designen und unter Verwendung angemessener Programmiertechniken die Kommunikation mit der Datenschicht implementieren.
  - Problemlösungen für konkrete Aufgabenstellungen analysieren sowie Programme selbständig entwerfen und mittels geeigneter Methoden moderner Softwaretechnologien realisieren
  - geeignete Entwicklungswerkzeuge und -systeme für eine Aufgabe auswählen und konfigurieren
  - Systeme modellieren und dokumentieren;
  - für die jeweilige Phase einer Softwareentwicklung die geeigneten Tests erkennen und beurteilen sowie Testfälle für konkrete Problemstellungen konzipieren und umsetzen;
  - nebenläufige Anwendungen auf Basis von Entwurfsmustern und Frameworks planen und entwickeln.
  - für die jeweilige Phase einer Softwareentwicklung die geeigneten Tests erkennen und beurteilen sowie Testfälle für konkrete Problemstellungen entwickeln

- Anwendungen auf Basis von Entwurfsmustern und Frameworks entwickeln
- für große Applikationen programmiertechnologische Konzepte ausarbeiten und Programmiervorgaben konzipieren.

Lehrstoff:

- Bereich Theoretische Informatik:
  - Zahlentheorie (Stellenwertsysteme, Konversionsalgorithmen, einfache Rechenoperationen in Fest- und Gleitkommaarithmetik, Grundzüge der Computernumerik, Fehleranalyse).
  - Algorithmen (Standardalgorithmen, Rekursion); Graphentheorie (Strukturen und Eigenschaften von Graphen, Speicherung von Graphen, Algorithmen in Graphen, Anwendungen und Problemlösungen).
  - Prädikatenlogik (prädikatenlogische Operatoren, Quantoren und Funktoren, Interpretationen und Modellbildung, mehrwertige Logik).
  - Algorithmen (Komplexität von Algorithmen, Optimierung und Anwendungen); Formale Sprachen (Metasprachen, Reguläre Ausdrücke, Grundlagen des Compilerbaus).
- Bereich Softwareentwicklung und -design:
  - Metasprachliche Problembeschreibung; Anweisungen, Kontrollstrukturen, Datentypen; Funktionen, Prozeduren, Methoden, Parameter, Rückgabewert; Integrierte Entwicklungsumgebungen, Teststrategien.
  - Anweisungen, Kontrollstrukturen; Skalare und zusammengesetzte Datentypen, Datenstrukturen; Funktionen, Prozeduren, Methoden; Basisalgorithmen.
  - Objektorientierte Programmierung; Polymorphie und Collections; Persistenz, Dateizugriffe, Datenbankzugriffe und Serialisierung; Speicherklassen und Speicherverwaltung.
  - Userinterfaces, Elemente graphischer Benutzeroberflächen, Eventhandling, Design, Layout, Usability; Design Patterns für verteilte Anwendungen; Statische und Dynamische Strukturen.
  - Modellierung, Softwarearchitektur, Design Patterns. Versionsverwaltung, Plug-ins, Bibliotheken, Dokumentationstools. Unit Tests, Erweiterte Teststrategien, Validierung. Prozesse, Threads, Kommunikation und Synchronisation.
  - Modellierung, Softwarearchitektur, Design Patterns. Aktuelle Trends der Softwareentwicklung und Programmiertechniken. Entwicklung von Anwendungen in Abstimmung mit fachtheoretischen Pflichtgegenständen.