TECHNISCHE UNIVERSITÄT DARMSTADT

Below you will find a tabular overview of the compulsory and core courses of the bachelor's degree in Computer Science at TU Darmstadt, once a short version and once including a short description of the learning content. Please add to the table in the right-hand column the successfully completed courses/modules of your previous degree programmes in which, in your view, equivalent content to the corresponding courses was provided. It is not necessary for admission that all listed courses have been covered in terms of content, but partially covered courses will only be counted partially. At least 60 ECTS are needed to fulfill the requirements.

Compulsory and core courses at TU Darmstadt	**Successfully completed courses with comparable content (Credits/Units)**
Functional and Object-oriented Programming Concepts	
Algorithms and Data Structures	
Digital Design	
Computer Organisation	
Parallel Programming	
Operating Systems	
Introduction to Compiler Construction	
Automata, Formal Languages and Decidability	
Propositional Logic and Predicate Logic	
Formal Methods in Software Design	
Computer Networks and Distributed Systems	
Computer Security	
Information Management	
Software Engineering	
Modeling, Specification and Semantics	
Visual Computing	
Introduction to Artificial Intelligence	
Probabilistic methods in computer science	
Scientific Computing	

Compulsory and core courses of the degree programme "Bachelor of Science in Computer Science" of the Department of Computer Science at TU Darmstadt	Successfully completed courses with equivalent content
Functional and Object-oriented Programming Concepts: ● Basic concepts of programming languages ● Foundations of functional programming languages ● Foundations of object-oriented programming languages ● Design and implementation of small software systems ● Basic type systems ● Fundamental data structures and algorithms and their complexity ● Recursion ● Simple I/O ● Basics of testing ● Documenting source code	
Algorithms and Data Structures: ● Data structures: array, list, binary search tree, B-tree, graph representation, hash table, heaps ● Algorithms: sorting algorithms, string matching, graph traversal, insertion, search, and deletion for data structures, shortest path search, minimal spanning trees ● Asymptotic complexity: run times, Big O notation, complexity classes P and NP, NP completeness ● Algorithmic strategies. for example: Divide-and-Conquer, dynamic programming, brute-force, greedy, backtracking, meta heuristics	
Digital Design: ● Digital Design: digital abstraction and its technological realization, number systems, logic gates, MOSFET transistors and CMOS gates, power consumption ● Combinational Logic Design: boolean equations and algebra, mapping equations to gates, multi-level logic circuits, four-valued logic (0,1,X,Z), logic minimization, combinational building blocks, timing	

• Sequential Logic Design: latches, flip-flops, synchronous logic design, finite-state machines, timing, parallelism • Hardware Description Languages: modeling of combinational and sequential circuits, structural modeling, modeling of finite-state machines, data types, parametrized modules, testbenches • Digital Building Blocks: arithmetic circuits, fixed-/floating-point representations, sequential building blocks, memory arrays, logic arrays	
Computer Organisation: • Architecture of Microprocessors: programming in assembly and machine language, addressing modes, tool flows, run-time environment • Microarchitecture: instruction set and architectural state, performance analysis, microarchitectures with single-cycle/multi-cycle/pipelined execution, exception handling, advanced microarchitectures • Memory and I/O-Systems: performance analysis, caches, virtual memory, I/O techniques, standard interfaces	
Parallel programming: • Foundations of parallel systems • Parallel architectures • Programming models for parallel computing • Parallel algorithms • Significant practical programming exercises covering the above topics • If necessary introduction to base programming languages	
• Operating Systems: • Introduction to Operating Systems (OS) - Role, purpose and design issues • Processes and Threads - OS structures, process control, abstractions, kernel/user modes and operations, context switching, interrupts • Inter-Process Communication - Message passing IPC, RPC, layers, interfaces, hierarchies	

TECHNISCHE UNIVERSITÄT DARMSTADT

• Coordination: Deadlocks - Process coordination, critical sections, deadlock characterization, deadlock detection and recovery, deadlock avoidance • Scheduling/Resource Management - Task ordering, preemptive and non-preemptive scheduling, schedulers and policies, OS implementations • Concurrency: Races, Mutual Exclusions - Critical sections, races, spin locks, synchronization • Programming Abstractions: Semaphores - Semaphores, Monitors • Memory Management - Storage structures, management/replacements approaches, virtual memory, paging, caching, segmentation • I/O - Device management, drivers, segmentation, interrupt handling, DMA • File systems - File systems requirements, design and implementation, file structures, directories, naming, partitions, virtual file systems • Fault Tolerance/Resilience - Fault types, fault handling approaches, reliable message delivery, OS reliability and availability, security issues • Embedded/RT OS - Memory/disk/performance management, recovery, fault-tolerances, real-time aspects • Distributed OS - Distributed computation and communication abstractions, synchronization, coordination, consistency • Virtual Machines - Purpose and types of virtualization, virtual file systems, Hypervisors	
Introduction to Compiler Construction: • Structure of compilers • Context-free grammars for the description of language syntax • Lexing and parsing techniques • Intermediate representations • Semantic analysis • Run-time organisation • Code generation • Software tools for compiler constructions • Implementation techniques for compilers	

Automata, Formal Languages and Decidability: • Introduction: transition systems, words, languages • Basic mathematical methods and proof patterns • Finite automata and regular languages, determinism and nondeterminism, closure properties and automata constructions, Kleene Theorem, Myhill-Nerode Theorem, pumping lemma • Grammars and the Chomsky hierarchy, context-free languages, pumping lemma, CYK algorithm; • Models of computation: PDA and Turing machines • Decidability and recursive enumerability in the Chomsky hierarchy	
Propositional Logic and Predicate Logic: • syntax and semantics of propositional logic, functional completeness and normal forms, compactness, complete proof calculi: resolution and a sequent calculus • Syntax and semantics of first-order logic, structures and assignments, normal forms, skolemization, Herbrand theorem, compactness, complete proof calculi: (ground) resolution and a sequent calculus, Gödel's Completeness Theorem • Undecidability of first-order logic; • optional: digressions on expressiveness and model checking	
Formal Methods in Software Design: • Modelling of concurrent software with the ProMeLa language • Formalisation of safety and liveness properties in propositional temporal logic • Theoretical Foundations of Model Checking • Verification of ProMeLa programs using the model checker SPIN • Syntax, semantics, and sequent calculus for typed first-order logic • Foundations of the contract-based software specification language JML • Dynamic logic as a first-order program logic • Formal software verification by symbolic execution and invariant reasoning	

• Tool-based verification of Java programs with the verification system KeY	
Computer Networks and Distributed Systems: • Foundations: Services, protocols, connection, layer model • Role of link layer, network layer, transport layer, application layer • Basic mechanisms (algorithms, protocols) for multiplexing, broadcast, multicast, routing and forwarding • Quality of service and reliability: definition and mechanisms • Coordination in distributed systems: from primitives to applications • Selected internet protocols and technology	
Computer Security: Part I: Cryptography • Background in mathematics for cryptography • Security objectives: Confidentiality, Integrity, Authenticity • Symmetric and asymmetric cryptography • Hash functions and digital signatures • Protocols for key distribution Part II: IT-Security and Dependability • Basic concepts of IT security • Authentication • Access control models and mechanisms • Basic concepts of network security • Basic concepts of software security • Basic concepts of web security • Dependable systems: error tolerance, redundancy, availability	
Information Management: Part 1: Structured data / databases Data Modeling: • Conceptual data models (ER / UML structure diagrams) • Conceptual design	

● Logical data model (relational model) ● Mapping from conceptual to logical model Relational query languages: ● SQL (in detail) ● Relational Algebra Database theory: ● Functional dependencies ● Design theory and normalization Implementation of database systems: ● Physical data storage ● Query processing and optimization ● Transaction processing Current trends in databases: ● Main-memory databases & Column-based data storage ● NoSQL databases ● Big Data Systems Part 2: Unstructured Data / Text Processing Basics of unstructured data: ● Storage and encoding of unstructured text ● Creating and annotating text corpora ● Lexical resources and knowledge bases Natural Language Processing: ● Segmentation ● Syntactic and semantic analysis ● Other Applications for unstructured data: ● Information Retrieval ● Information Extraction Advanced Topics: ● Introduction to research data management ● Data curation and visualization ● Documentation and archiving	
Software Engineering: ● Requirements Analysis ● Domain Modelling	

• Object-oriented Analysis and Design
• Software Architecture
• Software Quality, in particular:
    o Verification (among others, testing and static analysis)
    o Software Metrics
• Design Patterns
• Refactoring
• Software Evolution and Software Variability	

Modeling, Specification and Semantics: • Models and their significance for Computer Science • Introduction to discrete modeling using mathematical logic and algebraic concepts • Interpretation and faithfulness of formal models • Abstraction, refinement, composition, and decomposition of models • Systematic construction of models and deliberate design decisions • Syntax and operational semantics of programming languages • Introduction to specification languages • Syntax and denotational semantics of formal specification languages • Elementary proof techniques and their use • Modeling of systems and of requirements • Modeling of coordination and communication in concurrent systems	
Visual Computing: • Basics of perception • Basic Fourier transformation • Images, filtering, compression & processing • Basic object recognition • Geometric transformations • Basic 3D reconstruction • Surface and scene representations • Rendering algorithms • Color: Perception, spaces & models • Basic visualization	

Introduction to Artificial Intelligence Foundations: ● Introduction, History of AI ● Intelligent Agents Search: ● Uninformed Search ● Heuristic Search ● Local Search ● Constraint Satisfaction Problems ● Games: Adversarial Search Planning: ● Planning in State Space ● Planning in Plan Space Decisions under Uncertainty: ● Uncertainty and Probabilities ● Bayesian Networks ● Decision Making Machine Learning: ● Neural Networks ● Reinforcement Learning Philosophical Foundations	
Probabilistic methods in computer science: ● Basics from probability theory, statistics and information theory. ● Probabilistic approaches to graph-based modeling in computer science ● Basic probabilistic problems and use of probabilistic methods ○ in practical computer science (e.g. run-time analysis of programs, data compression), ○ in technical computer science (e.g., reliability of hardware, caching), and ○ in applied computer science (e.g., simulation of stochastic systems, probabilistic robotics). ● Selected randomized algorithms, their analysis by 'The Probabilistic Method', algorithms for automated decision making and optimization	

• Application of probabilistic methods in artificial intelligence (e.g. learning methods, neural networks) and data science • Implementation of probabilistic methods by means of practical programming examples	
Scientific Computing: • Fundamentals of scientific modeling and "The Scientific Method". • Modeling and system description using the example of mechanical systems • Problem specification for the simulation of complex models • Model building and identification using the example of mechanical systems • Model analysis of static systems by numerical methods for the solution of linear and nonlinear systems of equations • Model analysis and simulation of dynamic models by initial value problems with ordinary differential equations • Implementation of models and simulations using examples e.g. from robotics and other fields • Validation of models and simulations using measured data • Applications in the simulation and control of robots as well as physics-based animation and computer games	