Introduction to Computer Science

Dept. of Advanced Computing Sciences

BCS1110

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
A.J. Garnock-JonesA.R.K. Sai

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

The primary goal of Introduction to Computer Science is to introduce fundamental concepts and foster critical skills found throughout the field of computer science. Fundamental concepts include algorithms, computer architecture and hardware, models of computation, computer networks, and operating systems. Critical skills include abstraction, decomposition, pattern recognition, and algorithmic thinking. All concepts and skills are introduced in a lecture setting and explored further in the lab through the development of a wirelessly controlled microcontroller device. At the end of this course, students will appreciate the depth of the field and be prepared for subsequent research and educational activities.

Prerequisites

None.

Recommended reading

- "Computational Thinking for the Modern Problem Solver" by David Riley, Kenny A. Hunt
- "Computer Science Illuminated" by Nell B. Dale

Procedural Programming

Dept. of Advanced Computing Sciences

BCS1120

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
C. KouzinopoulosE. Hortal Quesada

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

The course provides the basics of computer science and computer programming. After a short introduction to computer organization and algorithmic thinking, the principles of programming are presented. The main topics of the course are: data types, variables, methods, parameters, decision structures, iteration and arrays. Programming skills will be acquired during practical sessions using the object-oriented programming language Java.

Prerequisites

None.

It appears as part of the pre-requisites of the second semester project in year 1, both projects of year 2, the year 2 course Databases and the year 3 courses, Parallel Programming and Robotics and Embedded Systems.

The course appears as desired prior knowledge for the courses Introduction to Objects in Programming, Data Structures and Algorithms, Software Engineering, Databases and Machine Learning.

Recommended reading

H. Schildt, Java: A Beginner's Guide, Eighth Edition, ISBN: 1260440214, McGraw-Hill Education

Discrete Mathematics

Dept. of Advanced Computing Sciences

BCS1130

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
O. D'HuysM. Musegaas

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

In this course, we build a mathematical framework that is based on logic and reason. The main objective of the course is to make students familiar with the language of mathematics. Students will learn how to make sound arguments and to detect where and why certain arguments go wrong. For this purpose, we will discuss the basic principles of logic and, closely related, the basic types of mathematical proofs. In doing so, we will encounter numbers such as integers, natural numbers and real numbers and we shall examine what makes these numbers special. After that, we will use basic logic to discuss, among other things, the following mathematical concepts: infinity, sets, relations, functions, permutations and combinations. Our fundamental tool in all of this is plain common sense. You really do not need your toolbox of mathematical formulas learned in previous studies and neither do you need a calculator. Pen and paper are the basic instruments needed. After completing each topic, exercises will be provided to be completed in class or at home, since mathematics is mainly learned by practising repeatedly.

Prerequisites

None.

Recommended reading

Discrete Mathematics by A. Chetwynd and P. Diggle (ISBN: 9780340610473)

Project 1-1

Dept. of Advanced Computing Sciences

BCS1300

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
6.0

Coordinator:
M. Boussé

Teaching methods:
Skills, Work in subgroups, Presentation(s), Project-Centered Learning

Assessment methods:
Participation, Assignment, Presentation and paper

Full course description

Students work on a project assignment in small groups of about six students. The group composition stays the same for the whole project and is announced shortly before the project opening in period 1.1. The students are guided through the project by a fixed tutor. The project assignment is divided into three subtasks (one per period) and is strongly related to the content of the courses from period 1.1 and 1.2. In period 1.1, after receiving the assignment for the whole project at the end of week 5, the students work full-time on the project in week 6. In this week, each group meets the tutor twice. In period 1.2, the students continue working on the project, while also having to attend the courses of that period. They meet their tutor approximately once a week. In period 1.3, the students work three weeks full-time on the project and meet their tutor about twice a week.

At the beginning of period 1.2 and 1.3, the students have to hand in a planning for the current phase. At the end of each period, the students have to give a presentation and the source code, presentation and an overview of who did what need to be uploaded to Canvas. While the presentations at the end of period 1.1 and 1.2 are in front of the examiners and the tutors, the presentations at the end of period 1.3 will additionally be in front of the fellow students. In period 1.3, they furthermore have to hand in a report and attend a product and report examination.

Project 1-1 will start in period 1.1 and period 1.2. The credits for the projects will become available at the end of period 1.3.

For each period, we will give a short explanation of the various parts. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching form, the schedule, and the examination method.

Prerequisites

This project has no prerequisites. This project occurs as part of the prerequisites of project 2-1.

Recommended reading

None

Objects in Programming

Dept. of Advanced Computing Sciences

BCS1220

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
D.J.N.J. SoemersT.H.J. Pepels

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course is a follow-up of the course Procedural Programming. It teaches object-oriented programming in Java. The main topics covered in the course are objects and classes, interfaces and polymorphism, event handling, inheritance, graphic user interfaces, exception handling, and streams.

Prerequisites

Desired prior knowledge: Basic Java Programming

Recommended reading

C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4 C. Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

Calculus

Dept. of Advanced Computing Sciences

BCS1440

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
G.M. SchoenmakersM. Musegaas

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

The following subjects will be discussed in Calculus: limits and continuity, differential calculus, integral calculus, and an introduction to sequences and series and multivariable calculus. In addition to the main facts and concepts, problem-solving strategies will be discussed. Both the intuition behind the concepts and their rigorous definitions will be presented along with simple examples of formal mathematical proofs.

Prerequisites

None

Recommended reading

University Calculus: Early Transcendentals, Global Edition (4th edition) by Joel Hass, Christopher Heil, Maurice D. Weir, and George B. Thomas (ISBN: 9781292317304).

Logic

Dept. of Advanced Computing Sciences

BCS1530

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
S.J. MaubachO. D'Huys

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course covers classical logical systems as propositional logic and first-order predicate logic, and it presents an introduction to dynamic logic and hoare logic. The course covers notation systems, syntax and semantics, valid consequences, semantic tableaux and natural deduction proof systems

Prerequisites

None

Recommended reading

"Logic in Action (Edition 2016) , Johan van Benthem, Hans van Ditmarsch, Jan van Eijck, Jan Jaspars" (www.logicinaction.org)

Linear Algebra

Dept. of Advanced Computing Sciences

BCS1410

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
P.W.L. DreesenM. Musegaas

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course introduces the fundamental concepts of linear algebra, and examines them from both an algebraic and a geometric point of view. First, we address what can be recognized without doubt as the most frequently occurring mathematical problem in practical applications: how to solve a system of linear equations. Then we discuss linear functions and mappings, which can be studied naturally from a geometric point of view. Vectors spaces are then introduced as a common framework that brings all themes together. Next, we shift from the geometric point of view to the dynamic perspective, where the focus is on the effects of iterations (i.e., the repeated application of a linear mapping). This involves a basic theory of eigenvalues and eigenvectors, which have many applications in various branches of science as for instance in problems involving dynamics and stability, in control theory, and in optimization problems found in data science. Key concepts in the course are vectors, matrices, systems of linear equations, eigenvalues, eigenvectors, linear transformations, and orthogonality. The software package Matlab is introduced in the accompanying computer classes, where emphasis is put on the application of linear algebra to solve real world problems.

Prerequisites

None.

Recommended reading

Linear Algebra and Its Applications (6th edition) by David C. Lay, Steven R. Lay, and Judi J. McDonald (ISBN: 978-1-292-35121-6).

Data Structures and Algorithms

Dept. of Advanced Computing Sciences

BCS1420

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
M.F.M. SondagF. Barile

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

As a continuation of the courses Procedural Programming and Objects in Programming, this course will treat the systematic design and application of data structures and algorithms. Data structures such as lists, trees, graphs, and strings, the associated algorithms and their complexity will be treated. Design principles for algorithms such as recursion, divide-and-conquer and dynamic programming will be treated as well.

Prerequisites

Desired Prior Knowledge : Discrete Mathematics, Procedural Programming and Objects in Programming. The course is desired prior knowledge for Theoretical Computer Science.

The course itself occurs as part of the pre-requisites of both projects of year 2 and the third year course Parallel Programming.

Recommended reading

Required Reading: Sedgewick and Wayne (2011) Algorithms Fourth Edition. Addison Wesley. ISBN: 978-0321573513

Recommended Reading: A Y Bhargava (2016). Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Manning. ISBN: 978-1617292231

Computer Architecture

Dept. of Advanced Computing Sciences

BCS1450

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
A.R.K. Sai

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

The goal of Computer Architecture is to provide a foundational understanding of how computers are built and operate at the hardware level. The course introduces digital logic, combinational and sequential circuits, hardware description languages, processor architecture, memory systems, and input/output systems. Students gain hands-on experience designing and simulating hardware components and will complete the course with a working understanding of the structure and function of a modern processor.

Prerequisites

None.

Desired Prior Knowledge : Introduction to Computer Science and Procedural Programmimg

Recommended reading

Computer Architecture: A Quantitative Approach (8th edition) by John L. Hennessy and David A. Patterson.

Project 1-2

Dept. of Advanced Computing Sciences

BCS1600

Semester 2:
26 Jan 2026 **19 Jun 2026**

Credits:
6.0

Coordinator:
X. ZhangO. D'Huys

Teaching methods:
Skills, Work in subgroups, Presentation(s), Project-Centered Learning

Assessment methods:
Participation, Assignment, Presentation and paper

Full course description

Students work on a project assignment in small groups of about six students. The group composition stays the same for the whole project and is announced before the project opening in period 1.4. The students are guided through the project by a fixed tutor. The project assignment is divided into three subtasks (one per period) and is strongly related to the content of the courses from period 1.4 and 1.5. In period 1.4, after receiving the assignment for the whole project at the end of week 5, the students work full-time on the project in week 6. In this week, each group meets the tutor twice. In period 1.5, the students continue working on the project, while also having to attend the courses of that period. They meet their tutor approximately once a week. In period 1.6, the students work three weeks full-time on the project and meet their tutor twice a week.
At the beginning of period 1.5 and 1.6, the students have to hand in a planning for the current phase. At the end of each period, the students have to give a presentation and the source code, presentation and an overview of who did what need to be uploaded to Canvas. While the presentations at the end of period 1.4 and 1.5 are in front of the examiners and the tutors, the presentations at the end of period 1.6 will additionally be in front of the fellow students. In period 1.6, they furthermore have to hand in a report and attend a product and report examination.
Project 1-2 will start in period 1.4 and period 1.5. The credits for the projects will become available at the end of period 1.6.
For each period, we will give a short explanation of the various parts. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching

form, the schedule, and the examination method.

Prerequisites

In order to participate in this project the student has to have passed two out of four courses from the set: Discrete Mathematics, Calculus, Procedural Programming and Objects in Programming.
This project occurs as part of the prerequisites of project 2-2.

Recommended reading

None.

Databases

Dept. of Advanced Computing Sciences

BCS1510

Period 5:
30 Mar 2026 **22 May 2026**

Credits:
4.0

Coordinator:
M.F.M. Sondag **K. Schneider**

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

This course introduces students to fundamental concepts of databases with a focus on relational database systems. Students will learn how to design, implement and optimize a relational database. The course provides a comprehensive study of data modeling, data description languages, and query facilities such as relational algebra and SQL. Students will also be exposed to the internal workings of database management systems with an overview of topics such as concurrency, recovery, indexing, and triggers. The course will cover current topics related to managing Big Data using alternate data models such as NoSQL. The students will apply their knowledge on these topics in designing and implementing a database system as a part of a group project.

Prerequisites

None.

Recommended reading

"Readings in Database Systems" by Peter Bailis, Joseph M. Hellerstein, Michael Stonebraker, 5th Edition.

Statistics

Dept. of Advanced Computing Sciences

BCS1520

Period 5:
30 Mar 2026 **22 May 2026**

Credits:
4.0

Coordinator:
A. Wodeyar

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Take home exam

Full course description

Statistics introduces the student to the main concepts of both probability theory and statistics. With respect to probability theory, students learn how to make use of random variables to extract the probability distribution of an experiment. Additionally, topics such as expectation, standard deviation, and independence will be discussed. The statistics part of the course discusses basic statistical topics such as the central limit theorem, verification of hypotheses, and confidence intervals. After completing this course students will have obtained an overview of commonly seen probability distributions, as well as several statistical procedures. Additionally, the student will be able to deal with problems that involve probabilities and determine an outcome for such problems (e.g., the expected outcome).

Prerequisites

None.

Recommended reading

None

Algorithmic Design

Dept. of Advanced Computing Sciences

BCS1540

Period 5:
30 Mar 2026 **22 May 2026**

Credits:
4.0

Coordinator:
D.O. MestelS.A. Chaplick

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

This course follows on from Data Structures and Algorithms, developing students' ability to design and (equally importantly) analyse algorithms for both correctness and efficiency. The course begins with a more in-depth treatment of greedy algorithms and dynamic programming (introduced in DSA), and how to analyse recursive algorithms with the master theorem. We then investigate how we can justify that a given problem is unlikely to have an efficient solution, via the notion of NP-completeness. Finally we cover some techniques we can use to attack NP-hard problems, namely backtracking, linear programming and branch-and-bound.

Prerequisites

Desired prior knowledge: Data Structures and Algorithms, Discrete Mathematics.

Recommended reading

Goodrich and Tamassia (2015) Algorithm Design and Applications. Wiley. ISBN: 978—1-118-33591-8

Introduction to Artificial Intelligence

Dept. of Advanced Computing Sciences

BCS2120

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
G. GoyalR. Cavill

Teaching methods:
Lecture(s), Skills

Assessment methods:
Written exam, Participation, Assignment

Full course description

The course starts with an analysis of the question “Can machines think?”, and the preconceptions usually encountered in discussions about that idea. Next, the metaphor of an “intelligent agent” is introduced, that is, of an entity that pursues goals by perceiving and acting flexibly and autonomously in a possibly very complex environment. Several state-of-the-art concepts, algorithms, and methods that enable computers (i.e., software and robots) to solve problems in a way that deserves to be called intelligent are discussed. Besides the technical background, societal and ethical issues around artificial intelligence such as the current quest for human-centered and explainable AI and how computational techniques might lead to biases and misconceptions through incautious data collection are discussed

Prerequisites

None.

Recommended reading

None

Intelligent User Interfaces

Dept. of Advanced Computing Sciences

BCS2130

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
K. ZarkogianniY.C. Semerci

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

Intelligent user interaction is a relatively new field in Computer Science, involving the areas of Human-Computer Interaction and Artificial Intelligence. It can often be seen as the intersection of computer science, behavioural sciences, and other field of study. This course will start with an overview of how to develop user-centric interfaces, what is the role of data analysis and research design methods in prototyping, and how different phases, from ideation to high fidelity prototyping relate to each other. Moreover, intelligent user interactions, involving technologies able to automatically recognize human intentions and behaviours, will be discussed. These can come, for example, from the fields of computer vision, speech processing, text mining. A thriving area in the field is that of interface personalization and affective computing, that is, computing driven by human emotions and behaviours.

Prerequisites

None.

Recommended reading

None

Operating Systems

Dept. of Advanced Computing Sciences

BCS2140

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
A.I. Iamnitchi

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

Description: This course introduces students to the core principles, design strategies, and implementation techniques of modern operating systems. It emphasizes both theoretical knowledge and hands-on practice, bridging the gap between hardware and software. Students will study process management, concurrency, scheduling, memory allocation, and file systems, while engaging in programming projects that simulate real-world system-level challenges. By the end of the course, students will have developed the ability to critically analyze operating system components, implement core mechanisms, and evaluate system performance and reliability.

Knowledge and understanding: By the end of the course, students will understand the role and structure of operating systems as intermediaries between hardware and applications, will be able to explain the principles of process management, describe different memory management techniques such as paging, segmentation, and virtual memory, and assess their trade-offs, and understand the structure and functionality of file systems and I/O subsystems.

Applying knowledge and understanding: Students will apply their knowledge through hands-on labs that utilize operating system APIs for process creation, synchronization, and communication, analyze and solve concurrency problems using synchronization primitives such as semaphores, mutexes, and monitors, and apply performance evaluation tools to assess the behavior of operating system modules and system-level programs.

Making judgements: Students will be required to critically evaluate the efficiency and fairness of scheduling algorithms and memory management strategies; assess trade-offs in operating system design decisions, balancing performance, usability, and security; analyze system failures, deadlocks, or bottlenecks, and propose appropriate solutions; reflect on security considerations in operating system design, especially regarding privacy, access control, and resource sharing.

Communication: Students will develop their communication skills through collaborative problem solving during the labs.

Learning skills: Students will develop skills that enable them to independently explore new operating system concepts and technologies beyond the course content and adapt their understanding of operating systems to emerging technologies, such as virtualization, distributed systems, and cloud computing.

Prerequisites

Desired Prio Knowledge:

Students should be comfortable with basic computer science concepts such as algorithms, data structures, and programming in languages like C, C++ or Java.

Recommended reading

“Operating Systems: Three Easy Pieces”, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. ISBN-10: 198508659X. ISBN-13: 978-1985086593

Computer Networks

Dept. of Advanced Computing Sciences

BCS2110

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
A.J. Garnock-Jones

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

This course introduces the fundamental concepts in computer networking. It covers the principles and structures of network architectures, protocols, and interfaces. Topics include the OSI and TCP/IP models, network devices, routing algorithms, wireless communication, network security, and emerging technologies.

Prerequisites

Desired Prior Knowledge : Operating Systems

Recommended reading

J. F. Kurose and K. W. Ross, Computer networking: a top-down approach, 8th edition. Hoboken, NJ: Pearson, 2021. ISBN 978-0-13-668155-7. (Alternative ISBN 978-1292405469 also acceptable; and if students have access to the 6th or 7th editions, this will also be acceptable)

Additional reading:

J. Day, Patterns in Network Architecture: A Return to Fundamentals. Prentice Hall, 2008. ISBN 0-13-225242-2.

Software Engineering and Architectures

Dept. of Advanced Computing Sciences

BCS2210

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
F. Barile

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

In this course, the student is introduced to the software engineering process. The course addresses the way in which large and complex software projects are conceived and managed. Topics in this course include, among others, requirement analysis, design methodologies, implementation strategies and test and maintenance procedures. In addition, the course discusses the software architectural design process. Several guidelines and several popular example software architectures are presented, as are different software delivery platforms and the current state of the art app development. After completing this course, the student will be able to judge the viability of a selected software development methodology and architectures.

Prerequisites

Desired Prior Knowledge : Procedural Programming, Objects in Programming, Object-Oriented Modeling.

Recommended reading

Ian Sommerville. 2015. Software Engineering (10th. ed.). Pearson. ISBN:978-0-13-394303-0

Roger Pressman. 2009. Software Engineering: A Practitioner ‘s Approach (7th. ed.). McGraw-Hill, Inc., USA. ISBN:978-0-07-337597-7

Embedded Programming

Dept. of Advanced Computing Sciences

BCS2410

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
C. Kouzinopoulos

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course introduces the fundamental concepts and practical skills required to develop software for embedded systems. Embedded systems are specialized computing devices designed to perform specific tasks within larger systems and are commonly found in industries such as automotive, healthcare, consumer electronics, robotics, digital agriculture and the Internet of Things (IoT). As part of this course, students will learn about microcontrollers, memory management, peripheral interfacing and power management techniques, using low-level programming languages such as C and assembly.

Topics include: microcontroller architecture, including CPUs, memory, timers, and hardware/software interrupts; use of C and assembly to write basic, low-level code for microcontrollers; interfacing with peripherals.

By the end of the course, students will be able to design and implement basic software for embedded systems.

Prerequisites

None.

Recommended reading

None

Computer Security

Dept. of Advanced Computing Sciences

BCS2420

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
A.R.K. Sai

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

Computer security is the process of detecting and preventing unauthorized and illicit access to a computer. As information systems have become mandatory in the commercial world, coupled with the increased frequency of security incidents, organizations now recognize the need for a comprehensive security strategy. This course will introduce a wide range of topics in such as security concepts and services, physical, operational, and organizational security, the role of people in systems security, an introduction to cryptography and the public key infrastructure, computing systems hardening, secure code, and secure applications development.

Prerequisites

None.

Recommended reading

None

Parallel Programming

Dept. of Advanced Computing Sciences

BCS2430

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
B. Küppers

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

Parallel programming is the paradigm of doing computations on a computer in parallel. This is possible, since nowadays almost all computer systems include so-called multi-core chips. To exploit the full performance of such systems, parallel programming needs to be employed.

This course covers the basics of parallel programming, such as multithreading, multiprocessing, and suitable data structures. Also,
prominent problems in parallel programming, such as deadlocks and race conditions, are discussed and solutions to these problems are presented.

Prerequisites

Desired prior knowledge: Procedural Programming (BCS1120), Objects in Programming (BCS1220), Data Structures and Algorithms (BCS1420), and Computer Architecture (BCS1450)

Recommended reading

Recommended literature: Eijkhout: The Art of HPC Vol. 1 & 2 (https://theartofhpc.com/)

Additional literature: Pacheco: An Introduction to Parallel Programming

Principles of Programming Languages

Dept. of Advanced Computing Sciences

BCS2220

Period 5:
30 Mar 2026 **22 May 2026**

Credits:
4.0

Coordinator:
A.J. Garnock-Jones

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

Programming Languages are ubiquitous. Every program with an “if” statement is, in an important sense, an interpreter for a programming language, however simple! This course will equip students with an analytical toolkit for understanding contemporary programming languages, relating them to a “standard model” of programming languages. The flip-side of analysis is construction: students will develop the basic understanding and knowledge necessary to begin creating programming languages of their own and writing tools which interpret, compile, or otherwise process other programs as their input data. On the way, students will examine the algebraic, functional “heart” of many programming languages, using equational reasoning to investigate time, change, concurrency and communication in programs.

Prerequisites

Discrete Mathematics; Procedural Programming; Object-Oriented Modelling.

Recommended reading

Krishnamurthi, Shriram. Programming Languages: Application and Interpretation. Version 3.2.2., 2023. https://www.plai.org/.

Additional literature:

1. Felleisen, Matthias, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. How to Design Programs. Second edition. MIT Press, 2014. https://htdp.org/.

2. Fisler, Kathi, Shriram Krishnamurthi, Benjamin S. Lerner, and Joe Gibbs Politz. A Data-Centric Introduction to Computing. Version 2023-02-21., 2023. https://dcic-world.org/.
3. Van Roy, Peter, and Seif Haridi. Concepts, Techniques and Models of Computer Programming. Cambridge, Massachusetts: MIT Press, 2004. https://www.info.ucl.ac.be/~pvr/book.html.
4. Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. 2nd ed. Cambridge, Massachusetts: The MIT Press, 1996. https://mitpress.mit.edu/sites/default/files/sicp/index.html.
5. Felleisen, Matthias, Robert Bruce Findler, and Matthew Flatt. Semantics Engineering with PLT Redex. Cambridge, Massachusetts: MIT Press, 2009.

Numerical Methods

Dept. of Advanced Computing Sciences

BCS2540

Period 5:
30 Mar 2026 **22 May 2026**

Credits:
4.0

Coordinator:
B. SakçakM. Boussé

Teaching methods:
Lecture(s), Project-Centered Learning

Assessment methods:
Written exam

Full course description

Numerical methods are techniques for solving problems from continuous mathematics (calculus and linear algebra) with the aid of a digital computer. In this course, we will cover the fundamental concepts of numerical mathematics, including the floating-point representation of real numbers, truncation and round-off errors, iterative methods and convergence. We will study the simplest and most important methods for core problems of continuous mathematics, namely the solution of algebraic equations and differential equations, interpolating data by polynomials, numerically estimating derivatives and integrals, approximating functions by polynomials and trigonometric series, solving systems of linear algebraic equations and computing eigenvalues. There will be a strong practical component, with students being expected to write their own numerical code and test the performance and suitability of different methods on various problems.

Prerequisites

Desired prior knowledge: Calculus, Linear Algebra

Recommended reading

Recommended literature: J.D. Faires & R. Burden, "Numerical Methods", International 4th Edition, Cengage, 2012; ISBN: 978-0-495-38569-1.

Additional literature: C.F. Gerald & P.O. Wheatley, "Applied Numerical Analysis", Seventh Edition, Pearson, 2003; ISBN: 0-321-13304-8. T. Siauw & A.M. Bayen, "An Introduction to Matlab Programming and Numerical Methods for Engineers", Academic Press, 2015; ISBN 978-0-12-520228-3.

M2-1: Intelligent Interaction

Dept. of Advanced Computing Sciences

BCS2710

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
10.0

Coordinator:
X. ZhangG. TangY.C. SemerciO. D'Huys

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Presentation, Assignment

Full course description

Elective Module Project 2-1 > **M2-1: Intelligent Interaction**

Course: Image and Video Processing (period 2) +

Project: Project 2-1 Human-Computer Interaction (semester 3, bachelor year 2)

Project and module coordinator:

- X. Zhang

Course coordinators:

- G. Tang

Each elective module comprises an elective course, an elective project, and related skill classes. Second year students **choose one** out **of the two** electives (Module Project 2-1) during semester 3.

Course description: In this course, students explore core image and video processing algorithms alongside modern deep-learning approaches. We begin by linking cameras to the physical world through camera calibration and geometry. We then cover color-space fundamentals and their connection to human visual perception, followed by spatial-domain techniques, including linear and non-linear filtering, segmentation, denoising, and smoothing. Frequency-domain transforms (Fourier, DCT) are introduced for enhancement, de-noising, and restoration, including practical insight into

standards such as JPEG. The course extends to video analysis with motion estimation and its role in compression (e.g., MPEG). In the second half, we focus on recent advances in deep learning for vision. Students study the workings of convolutional neural networks, vision transformers, and recurrent models, and apply them to tasks such as object detection, segmentation, enhancement, and object tracking. Hands-on lab sessions accompany the lectures throughout the semester, where students implement algorithms, run controlled experiments, and reproduce results from recent image and video processing research papers to build both practical skill and critical insight. By the end of the course, students will be able to design, implement, evaluate, and clearly communicate complete image/video processing pipelines, classical and deep-learning-based, on real datasets.

Assessment Method: Labs, Assignments, Reports

Project Description:
Project 2-1 Human-Computer Interaction: The central topic of this project is to develop and to evaluate a real-time human-computer interface using the parallax effect (https://en.wikipedia.org/wiki/Parallax) to generate the illusion of a 3-dimensional interactive scene. The project lets you practice the development of software that must run efficiently on decent computer hardware as well as the generation of interactive computer graphics that must react in real-time. You further practice the use of one of the most popular computer vision libraries: OpenCV, and apply your skills in performing user studies to evaluate your self-developed human-computer interface. You will use GitHub to host your code and effectively collaborate with your teammates, and end up with a project that you can proudly display on your GitHub profile as part of your portfolio of programming work. You are explicitly allowed to choose from a number of programming languages including Java, Python, C++ and computer graphics libraries.

Prerequisites

Students must have passed Project 1-1.

Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms.

The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 1. This project is a prerequisite for Project 3-1.

Recommended reading

None

M2-1: AI and Machine Learning

Dept. of Advanced Computing Sciences

BCS2720

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
10.0

Coordinator:
X. ZhangP. BosiljE.N. SmirnovE. Hortal Quesada

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Presentation

Full course description

Elective Module Project 2-1 > **M2-1: Artificial Intelligence and Machine Learning**

Course: Machine Learning (period 2) +

Project: Project 2-1 Adaptive Systems (semester 3, bachelor year 2)

Project and module coordinator:

- X. Zhang

Course coordinators:

- E. Hortal Quesada
- E.N. Smirnov
- P. Bosilj

Each elective module comprises an elective course, an elective project, and related skill classes.

Second year students **choose one** out **of the two** electives (Module Project 2-1) during semester 3.

- Course Machine Learning (period 2): Machine learning is a major frontier field of artificial intelligence. It deals with developing computer systems that autonomously analyse data and automatically improve their performance with experience. This course presents basic and state-of-the-art techniques of machine learning. Presented techniques for automatic data classification, data clustering, data prediction, and learning include Decision Trees, Bayesian Learning, Linear and Logistic Regression, Recommender Systems, Artificial Neural Networks, Support Vector Machines, Instance-based Learning, Rule Induction and Clustering.

Lectures and practical assignments emphasize the practical use of the presented techniques and prepare students for developing real-world machine-learning applications.

- Project 2-1 Adaptive systems: In this project, you will use Artificial Intelligence (AI) and Machine Learning (ML) techniques to control autonomous agents in a real-time video game engine. There will be a strong emphasis on combining and using existing libraries and tools (e.g., a video game engine, and existing implementations of AI and ML techniques), and building your own new code around it. You will use GitHub to host your code and effectively collaborate with your teammates, include clear documentation (for yourselves and others) on how to correctly install any dependencies, and end up with a project that you can proudly display on your GitHub profile as part of your portfolio of programming work. The used programming languages will be C# and Python.

Prerequisites

Prerequisites: Students must have passed Project 1-1.

Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms.

The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 1. This project is a prerequisite for Project 3-1.

Recommended reading

None

DACS Honours Programme - MaRBLe 2.0 (2-1)

Dept. of Advanced Computing Sciences

KEN2320

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
6.0

Coordinator:
R. Cavill

Teaching methods:
Assignment(s), Working visit(s)

Assessment methods:
Final paper

Full course description

During MaRBLe 2.0 you will get the opportunity to work on a state-of-the-art research project. Work will be organized in a similar way as in professional research institutes where participants work together as individual experts on a team project. Participation is open to excellent and motivated students.

Prerequisites

None.

Recommended reading

None.

M2-2: High Performance Computing

Dept. of Advanced Computing Sciences

BCS2730

Semester 2:
26 Jan 2026 **19 Jun 2026**

Credits:
10.0

Coordinator:
K. SchneiderC. Kouzinopoulos

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Participation, Presentation and paper

Full course description

Elective Module Project 2-2 > **M2-2: High Performance Computing**

Course: High Performance Computing (period 5) +

Project: Project 2-2 High Performance Computing (semester 4, bachelor year 2)

Each elective module comprises an *elective course* , an *elective project* , and related *skill classes* .

Second year students **choose one** out **of the two** electives (Module Project 2-2) during semester 4.

Project and module coordinator:

- K. Schneider

Course coordinators:

- C. Kouzinopoulis

Course description:

This course introduces the fundamental concepts and practical skills required to develop High Performance Computing (HPC) applications. It covers fundamental principles, architectures, and programming models for modern parallel and distributed systems. As part of this course students will explore key HPC concepts, including parallel computing paradigms, memory hierarchies, optimization techniques, and workload distribution across CPUs, GPUs, and accelerators. The course includes

hands-on experience with parallel programming frameworks such as MPI, OpenMP, and CUDA, enabling development of efficient and scalable applications.
By the end of the course students will understand HPC architectures and parallel programming models, gain experience with HPC tools and gain the ability to develop and optimize parallel applications for multi-core and distributed systems.

Project description:
In this project, students individually indicate their preferences for a given umbrella project. A least regret algorithm ensures that students will be allocated to a group of 6-7 students corresponding to an umbrella project such that the overall regret is minimal. In their project groups, the students come up with a project topic fitting to their umbrella topic in co-creation with their tutor and the examiners.
The project is divided into three phases. In the first phase, the groups define the topic and the project scope. At the end of that phase, they hand in a project plan and a prototype. The second phase is the implementation phase. Students should aim for a first version of the product that already allows for meaningful experiments. At the end of that phase, the students meet with the examiners in a formative assessment moment to discuss their progress and future plans. The third phase is used to enhance the product, write the project report and to create a poster that will be presented in a poster fair at the end of the project. The students' understanding of the project is tested in a project defense.
During the project phases, each group is guided by a tutor. The students work part-time in phases 1 and 2 of the project (with a full-time project week in-between), while they work full-time on the project in phase 3.

Assessment methods:
Course: Written exam
Project: Poster Presentation, Project Defense, Online submissions, Project meetings

Prerequisites

Prerequisites: Basic knowledge of programming, data structures, and algorithms. Familiarity with Linux is beneficial but not required

Students must have passed Project 1-2.

Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms.

The student needs to be registered for or has already completed at least three courses of the programme in year 2, semester 2. This project is not a prerequisite for another project / course.

Recommended reading

None

M2-2: Cybersecurity

Dept. of Advanced Computing Sciences

BCS2740

Semester 2:
26 Jan 2026 **19 Jun 2026**

Credits:
10.0

Coordinator:
K. SchneiderB. Küppers

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Participation, Presentation

Full course description

Elective Module Project 2-2 > **M2-2: Cybersecurity**

Course: Information Security (period 5) +

Project: Project 2-2 Cybersecurity & IoT (semester 4, bachelor year 2)

Project and module coordinator:

- K. Schneider

Course coordinators:

- B. Küppers

Course Description

Information security is a subfield of Computer security dealing with protecting information, which means to ensure confidentiality, integrity, and availability of information. Topics such as Cryptography, Access Control, Identification and Authentication play a key role in this field to ensure the technical foundation for information security. However, nowadays, where information is in constant flow across the globe, network security and (global) privacy laws are topics that also need to be considered when ensuring information security.

The course Information Security focuses on topics related to the secure storage and transmission of information. The need for these measures as well as legal requirements are discussed and

technological and organizational ways of ensuring compliance with needs and regulations are discussed. The course covers a wide variety of topics, such as cryptography, infrastructures, and privacy enhancing technologies. The lectures are accompanied by labs that will help the students to understand the concepts discussed in the lectures.

Project description:
In this project, students individually indicate their preferences for a given umbrella project. A least regret algorithm ensures that students will be allocated to a group of 6-7 students corresponding to an umbrella project such that the overall regret is minimal. In their project groups, the students come up with a project topic fitting to their umbrella topic in co-creation with their tutor and the examiners.
The project is divided into three phases. In the first phase, the groups define the topic and the project scope. At the end of that phase, they hand in a project plan and a prototype. The second phase is the implementation phase. Students should aim for a first version of the product that already allows for meaningful experiments. At the end of that phase, the students meet with the examiners in a formative assessment moment to discuss their progress and future plans. The third phase is used to enhance the product, write the project report and to create a poster that will be presented in a poster fair at the end of the project. The students' understanding of the project is tested in a project defense.
During the project phases, each group is guided by a tutor. The students work part-time in phases 1 and 2 of the project (with a full-time project week in-between), while they work full-time on the project in phase 3.

Assessment methods:
Course: Written exam
Project: Poster Presentation, Project Defense, Online submissions, Project meetings

Each elective module comprises an elective course, an elective project, and related skill classes. Second year students **choose one** out **of the two** electives (Module Project 2-2) during semester 4.

Prerequisites

Prerequisites: Students must have passed Project 1-2.

Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms.

The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 2. This project is not a prerequisite for another project / course.

Recommended reading

None

DACS Honours Programme - MaRBLe 2.0 (2-2)

Dept. of Advanced Computing Sciences

KEN2620

Semester 2:
26 Jan 2026 **19 Jun 2026**

Credits:
6.0

Coordinator:
R. Cavill

Teaching methods:
Assignment(s), Working visit(s)

Assessment methods:
Final paper

Full course description

In MaRBle 2.0, you will get the opportunity to work on a state-of-the-art research project. Work will be organized in a similar way as in professional research institutes where participants work together as individual experts on a team project. MaRBle takes place in year two of the bachelor's programme.

Selection of honours students will happen in the second semester of year 1. If you successfully complete the honours programme, this will be certified on an honour's diploma supplement.

Recommended reading

None.

Graph Theory

Dept. of Advanced Computing Sciences

BCS3110

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
M. Mihalak

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

A graph is simply a collection of objects and a binary relation between the objects. Visually, it is a bunch of points, some of which are joined by lines.
This deceptively simple structure is one of the cornerstones of both theoretical and applied computer science. A great many problems that arise in the real world can be modeled as graph problems. Several classical examples include the problem of finding the shortest route between two cities, of maximizing flow in a network of pipelines, or of finding an optimal pairing between producers and consumers. In this course we will look at both the algorithmic/applied side of graph theory and its more abstract mathematical foundations, because the latter is often important for understanding the former. We will cover topics such as paths, tours, trees, matchings, flows, colorings, and connectivity.

Prerequisites

Desired Prior Knowledge : Discrete Mathematics; Data Structures and Algorithms

Recommended reading

None.

Digital Society

Dept. of Advanced Computing Sciences

BCS3111

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
M.B. Archer

Teaching methods:
Project-Centered Learning

Assessment methods:
Final paper, Participation, Presentation, Oral exam

Full course description

Digital technologies affect our lives in increasingly profound and inescapable ways. Digitalization has changed the way we interact with friends, family members, and romantic partners. It has changed the way we access information and, subsequently, the kind of information we are able to access. Digital technologies have impacted trust in institutions and have precipitated new societal expectations regarding the relationship between governments and constituents. At the same time, it is important not to attribute these changes solely to the introduction of new technologies, but to instead understand these technologies as situated in specific social, economic, and political contexts.

The phrase ‘digital society’ attunes us to the interplay between the digital and the social, challenging the boundary between these two domains, similar to the way the notion of socio-ecology challenges the boundary between society and the environment. The goal of this 4 ECTS course is to equip students with a conceptual vocabulary to critically analyze the impacts of digitalization on and within society and the environment, and vice versa. Through a combination of lectures and discussion seminars, we will critically examine some of these entanglements through a selection of case studies and examples, from e-waste to ChatGPT to surveillance technologies.

Learning objectives
1. Knowledge and understanding: Supported by advanced texts, students develop a systematic understanding of new insights and methods in the academic field studying the relation between

digital technology and social aspects, the digital economy, privacy, surveillance, fakeness and socio-ethics.
2. Applying knowledge and understanding: Students can devise and sustain academic arguments pertaining to the relation between digitalization and society. Capable of conceiving, designing, and conducting a substantial process of research on platformization and datafication.
3. Making informed judgments: Students will be able to gather and interpret relevant data with the aim to further sharpen their ability to critically analyze, evaluate, and synthesize new and complex ideas from the academic field studying the relation between digital technology and social conditions, the digital economy, privacy, surveillance, fakeness, and socio-ethics.
4. Communication: Students fortify their capacity to communicate ideas, problems, and solutions pertaining to (the effects of) digitalization, datafication, platformisation in society with (a) peers, (b) the larger scholarly community, (c) non-IT- experts and (d) society in general. They develop a sense of multi-disciplinary group-work and learn to communicate and collaborate across disciplinary borders.
5. Learning skills: This course hands students the contexts, tools, insights, and techniques to help them to continue their studies of the complex relations between socio-technical developments (e.g. digitalization, datafication, platformisation) and societal outcomes (e.g. disintermediation, data colonialism, ethics) independently and autonomously.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

This is a shared course (maximum capacity of 60 students).

Prerequisites

None

Recommended reading

None

Ubiquitous Comp. & Internet of Things

Dept. of Advanced Computing Sciences

BCS3120

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
G. Tang

Teaching methods:
Project-Centered Learning

Assessment methods:
Assignment

Full course description

Ubiquitous Computing imagines micro-sized processors, sensors, and networks invisibly integrate into daily objects. IoT realizes this by linking smart devices into edge-intelligent ecosystems. This course teaches you to design, build, and optimize such AI-powered systems. The course will be divided into two major components: the basics of IoT systems, and the advances in efficient machine learning systems for edge AI and IoT.

The course will start with the basics of IoT systems that dive into the fundamentals of low-power computing hardware, real-time operating systems, and the communication technologies, like BLE, Wi-Fi, cellular, and more, that let tiny devices communicate reliably under tight energy budgets. Next, you will learn the sensing and perception, on how digital MEMS sensors work, how environmental monitors, optical biosensors and micro-cameras can be interfaced.

After the basics, the second half of the course shifts focus to edge AI systems. You will not only learn how to use deep learning methods to solve real-world problems, but you will also learn how to balance energy efficiency, memory cost, and latency in resource constrained systems for the real-world context. Signal-processing techniques flow naturally into the emerging field of edge AI and Tiny ML, where you will train, compress and deploy neural-network models that run with kilobytes of RAM to meet hard latency, memory, and battery-life constraints. State-of-the-art efficient

machine learning systems will be introduced, including AI accelerators, neuromorphic processors, and advanced efficient deep learning algorithms.

The course will have weekly hands-on labs on STM32U5 development boards, presentations on state-of-the-art research papers, plus a course project that challenges you to address a real-world problem, whether it's a posture-aware wearable, a vision-enhanced recycling bin or a smart air-quality sentinel. Alongside technical deliverables you will document trade-offs, justify architectural decisions and pitch your prototype in a final video showcase. By the end of the course you will be able to architect complete IoT pipelines, deploy efficient edge AI models and communicate your solutions with the clarity.

Prerequisites

None

Recommended reading

None

Game Theory

Dept. of Advanced Computing Sciences

BCS3130

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
G.M. Schoenmakers

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

We introduce the field of Game Theory. Game Theory is the mathematical study of problems, called games, that involve two or more decision makers, called players, who each have their own individual preferences over the possible outcomes. In a game, each player always aims to maximize his individual payoff and chooses his actions accordingly. These actions may be probabilistic or deterministic, depending on the situation. Meanwhile he reasons logically about actions that might be taken by the other players. A basic difference exists between strategic and non-strategic models. Both types of models and their solution concepts will be discussed. Issues like value, fairness, manipulations, threats, optimality and rationality will be addressed.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

Prerequisites

Discrete Mathematics, Linear Algebra

Recommended reading

None.

Robotics and Embedded Systems

Dept. of Advanced Computing Sciences

BCS3236

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
B. SakçakR. Möckel

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

Nowadays, a variety of products require that algorithms from data science and artificial intelligence are adapted to and implemented in robotic and embedded systems. Applications that heavily rely on intelligent robotic and embedded systems include self-driving cars, autonomous drones, intelligent industrial robots in (semi-) autonomous factories, smart phones, intelligent medical devices, and distributed intelligent embedded devices in smart homes.

In this course, students receive an introduction to the fields of robotics, embedded systems, and real-time control. Students obtain an overview of state-of-the-art intelligent robotic and embedded systems in academia and industries. Students gain hands on experience in programming embedded robotic systems using embedded processors and a modular robotic system developed at DACS. Students learn about communication standards for embedded systems, sensors, and actuators. Student practise and strengthen their expertise in data science and knowledge engineering by applying mathematical methods for controlling robotic systems: They study control techniques including PID control, forward and inverse kinematics as well as locomotion control and learning using central pattern generators. The course concludes with a robot competition where students build and program robots using a modular robotic system.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

Maximum number of 130 students can follow this course. (Shared with the other bachelor)

Prerequisites

Procedural Programming and Objects in Programming.

Desired prior knowledge: Calculus, Linear Algebra, Machine Learning.

Recommended reading

None

Introduction to Quantum Computing

Dept. of Advanced Computing Sciences

BCS3241

Period 1:
1 Sep 2025 **24 Oct 2025**

Credits:
4.0

Coordinator:
D. Dibenedetto

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course offers an introduction to the interdisciplinary field of quantum computation. The focus will lie on an accessible introduction to the elementary concepts of quantum mechanics, followed by introducing the mathematical formalism and a comparison between computer science and information science in the quantum domain. The theoretical capability of quantum computers will be illustrated by analysing fundamental algorithms of quantum computation and its potential applications.

Quantum technology has become one of the most prominent interdisciplinary fields of recent research. This course will focus on introducing the mathematical concepts underpinning quantum computation, and on explaining how this new computational paradigm might potentially offer possibilities beyond the scope of conventional computers. Topics that will be introduced and discussed include:

1. most common models of quantum computation (e.g., quantum circuits and measurement-based quantum computing).
2. An exposition of the machinery borrowed from quantum mechanics, such as superposition of states, quantum entanglement, (de)coherence etc., which gives rise to the potential speed-up of quantum algorithms over their classical analogs.
3. Some of the most common quantum algorithms (searching, factoring etc.) and protocols (quantum teleportation, EPR paradox). The course will finish with an exposition of potential applications of quantum computation and algorithms in other fields (such as security/cryptography, AI, optimization etc.)

Important: no prior knowledge in quantum mechanics is assumed or required, and all necessary concepts will be introduced and motivated from a mathematical and theoretical computer science point of view. Possible quantum architectures and/or related hardware issues will not be discussed.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

Prerequisites

Linear Algebra.

Recommended reading

- Isaac Chuang, Michael Nielsen, “Quantum Computation and Quantum Information”, 10th Anniversary Edition, Cambridge University Press, 2011.
- N. David Mermin, “Quantum Computer Science: An Introduction”, 1st Edition, Cambridge University Press, 2007

Project 3-1

Dept. of Advanced Computing Sciences

BCS3300

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
6.0

Coordinator:
R. MöckelK. Schneider

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Participation, Assignment, Presentation and paper

Full course description

The students in project 3-1 will work on project tasks of diverse fitting to their curriculum provided by (external) customers. The project will be carried out by small groups of students, which will be guided by a tutor. Company supervisors or staff members of the Department of Advanced Computing Sciences act as customers of the projects. Additionally, two examiners of the Department of Advanced Computing Sciences assess the students' achievements. The group composition will be based on the students' preferences.

The project is split up into three phases coinciding with the three periods of the first semester in year 3:

In the first phase, students will become familiar with the project assignment, and lay a foundation for the next phases by scoping the project well. The students meet their customers and discuss the framework, requirements, and deliverables of the project. Based on the outcome of this discussion and on research on the project topic, the students create a project plan including, e.g., a risk analysis, a literature review, and a time planning. Last but not least, a prototype needs to be submitted at the end of phase 1.

The aim of phase 2 is to implement a solid product that can further be enhanced in phase 3. More advanced concepts will be implemented and a planning for phase three is made. In the midway

evaluation at the end of phase 2, the experimental results should help the students to convince their customers that the project can successfully be completed in phase 3.

Phase 3 consists of three full-time weeks, in which the students improve their product based on their ambitions and agreements with the customer. They have to document their work in the form of a scientific report and a business presentation. Furthermore, the students have to defend their choices and implementations in a project defense with the DACS examiners.

The project will be accompanied by so-called skills classes. These classes help the student develop competencies that are useful in this project and their further careers.

Project 3-1 will start in period 3.1 and period 3.2 with weekly meetings. The credits for the projects will become available at the end of period 3.3.

Prerequisites

Pre-requisites: Successfully passed project 2-1

Recommended reading

None.

DACS Honours Programme - CS@Work (3-1)

Dept. of Advanced Computing Sciences

BCS3310

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
6.0

Coordinator:
K. Driessens

Teaching methods:
Assignment(s), Working visit(s)

Assessment methods:
Final paper

Full course description

The DACS Honours Programme consists of KnowledgeEngineering@Work (KE@Work), Computer Science@Work (CS@Work) and (MaRBLe 2.0).

Students admitted to the CS@Work path are placed at a company or organization in the region through a careful selection and matching process. During the third year of the bachelor's programme, they spend 50% of the time in class and 50% at the company, where they work on solving academic challenges and complex business problems, under supervision of dedicated business and DACS supervisors

DACS Honours Programme - MaRBLe 2.0 (3-1)

Dept. of Advanced Computing Sciences

KEN3320

Semester 1:
1 Sep 2025 **23 Jan 2026**

Credits:
6.0

Coordinator:
R. Cavill

Teaching methods:
Assignment(s), Working visit(s)

Assessment methods:
Final paper

Full course description

In MaRBle 2.0 (3-1), you will get the opportunity to work on a state-of-the-art research project. Work will be organized in a similar way as in professional research institutes where participants work together as individual experts on a team project. MaRBle takes place in year two of the bachelor's programme.

Selection of honours students will happen in the second semester of year 1. If you successfully complete the honours programme, this will be certified on an honour's diploma supplement.

Prerequisites

Eligibility requirements for MaRBLe 2.0 (3-1) entail that students:
• have passed all courses/components in the first year of the Bachelor's programme at first opportunity;
• have obtained a GPA of at least 7.5 for all courses of year 1 (to be eligible for pre-selection a GPA of at least 7.5 has to be obtained for blocks 1 to 4);
• have not been convicted of fraud and have not been reprimanded for a violation of house rules or code of conduct.

Recommended reading

None.

Study Abroad

Dept. of Advanced Computing Sciences

KEN3600

Semester 1:
1 Sep 2025 **23 Jan 2026**

Semester 2:
26 Jan 2026 **19 Jun 2026**

Credits:
30.0

Coordinator:
M. Musegaas

Assessment methods:
Written exam, Attendance, Assignment

Full course description

DACS offers its students the possibility to study a semester abroad at one of DACS partner universities. Third year bachelor's students and 2nd year master's students can get the opportunity to study a semester abroad, as part of their education programme in Maastricht. The credits received abroad will be transferred / part of your programme at DACS in Maastricht.Of course, this is only possible after approval of the Board of Examiners. There are several universities where DACS can send its students to.

If you still have questions afterwards you may contact our Exchange Coordinator Luc Giezenaar via: dacs-iro@maastrichtuniversity.nl

Prerequisites

You have to obtained at least 40 ECTS of year 1 courses.

Software and Systems Verification

Dept. of Advanced Computing Sciences

BCS3150

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
P.J. Collins

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

Have you ever written a program with a bug in it? Then this course is for you! Software verification tools can check whether your program works by showing that it correctly satisfies its specification, or finds a case in which it can go wrong. Unlike unit testing and other software validation methods, verification tools use formal methods to rigorously prove correctness. Similar techniques can be used to show that (mathematical models of) cyber-physical systems work as designed.
In this course, we will start by and introducing the main notions of object-oriented program verification, including pre- and post-conditions for methods, and class invariants. We shall use Hoare logic to convert programs and their specifications into logical statements to be proved. We shall apply these techniques to the verification of simple programs written in Java.
In the second part of the course, we consider formal models of software and systems as labelled transition systems (automata), using temporal logics for specification, and consider the fundamental algorithms for verification. We shall apply these algorithms to simple discrete verification problems, such as vending machines and communications systems, modelled using a specification language such as SMV. Finally, we will look at simple continuous systems, such as robots and electronic systems, and show how to verify these using rigorous numerical methods based on interval arithmetic.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

Prerequisites

Desired prior knowledge : Logic (BCS1530), Principles of Programming Languages (BCS2220)

Recommended reading

J.B. Almeida, M.J. Frade, J.S. Pinto & S. Melo de Sousa, “Rigorous Software Development: an Introduction to Program Verification”, Springer, 2011.

C. Baier & J.P. Katoen, “Principles of Model Checking”, MIT Press, 2008.

L. Jaulin, M. Kieffer, O. Didrit & E. Walter, "Applied Interval Analysis", Springer, 2001.

Block Chains

Dept. of Advanced Computing Sciences

BCS3210

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
V. Urovi

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course provides a general introduction to the fundamental principles, technologies, and applications of blockchain systems. The objective is for students to gain a foundational understanding of blockchain architecture, cryptographic building blocks, consensus mechanisms, and decentralized applications (DApps). The course emphasizes both theoretical concepts and practical skills, equipping students to understand research in the blockchain space and begin applying core techniques to technical projects and prototypes.

Topics include the design and implementation of simple blockchain systems, cryptographic foundations such as hashing and digital signatures, consensus algorithms (e.g., Proof of Work), smart contract development on Ethereum, and exploration of modern applications like NFTs, decentralized finance (DeFi), and DAOs (Decentralized Autonomous Organizations). The course consists of lectures to understand the theoretical foundations, followed by hands-on computer sessions centered around practical applications of these concepts. During practicals, students build and explore real blockchain components through guided programming exercises. By applying the techniques on case studies, students develop a thorough understanding of blockchain technologies and their use.

Maximum number of 60 students can follow this course.

Prerequisites

None.

Desired Prior Knowledge: Programming, Python.

Recommended reading

None

Startup Engineering: Building Scalable Tech Ventures

Dept. of Advanced Computing Sciences

BCS3220

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
R.P.M.G. Broersma

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

The **Startup Engineering: Building Scalable Tech Ventures** course provides an interdisciplinary approach to entrepreneurship and computer science, equipping students with the business acumen needed to build and scale technology-driven startups. The course covers fundamental concepts in customer development, business modeling, product-market fit, lean startup methodology, and scalable infrastructure. Students will gain practical experience by designing and prototyping their own startup ideas, leveraging modern tools and frameworks. Emphasis is placed on business decision-making, entrepreneurial strategy, and real-world case studies of successful tech startups.

Prerequisites

None.

Desired Prior Knowledge:
The course serves as foundational knowledge for advanced entrepreneurship and software engineering courses.

Recommended reading

None

Cryptography

Dept. of Advanced Computing Sciences

BCS3230

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
B.J.M. Mennink

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

This course provides a broad introduction to the fundamental principles and techniques of modern applied cryptography. The objective is that students will gain a solid understanding of cryptographic concepts at both an algorithmic and theoretical level and will be able to critically engage with technical literature and apply core methods in practice. The course covers advanced cryptographic topics such as secure encryption and authentication methods, public-key and post-quantum cryptography (including digital signatures and key exchange), and real-world protocols like secure messaging and TLS.
It also introduces important security definitions and proof techniques, including notions like existential forgery, zero-knowledge proofs, and challenge-response protocols.

Lectures focus on presenting the mathematical and algorithmic basis of cryptographic techniques, while practicals will allow students to implement and experiment with them in realistic scenarios. Through a combination of theoretical study and hands-on exercises, students develop both a conceptual and practical grasp of cryptography.

Prerequisites

None.

Desired Prior Knowledge: Discrete Mathematics, Linear Algebra, Procedural Programming.

Recommended reading

None

Large Scale IT and Cloud Computing

Dept. of Advanced Computing Sciences

BCS3239

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
M. Politze

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

The course offers a comprehensive introduction to the field of scalable IT systems, so-called "Big IT", and cloud computing. After a technical introduction to the available methodologies of setting up and running scalable systems, use cases are presented. These use cases emphasize the correlation of the processes and requirements of large institutions and possible technical solutions. A special focus is put upon the question which technological platform is best used for which use case as well as process aspects of scaling. Security aspects specific to cloud computing are discussed along the use cases. Cloud computing, as a special case of scalable IT, is discussed in detail. Different cloud providers are presented and evaluated in the context of university requirements, i.e. requirements posed by research and teaching processes.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

A maximum number of 120 students can take this course. (shared course with the other bachelor)

Prerequisites

None

Recommended reading

None

Event-based Vision

Dept. of Advanced Computing Sciences

BCS3240

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
P. BosiljG. Goyal

Teaching methods:
Lecture(s), Skills

Assessment methods:
Written exam, Assignment

Full course description

Event cameras are novel visual sensors that work quite differently from conventional RGB cameras. Instead of taking pictures at fixed times, they only capture changes in brightness as they happen asynchronously. This bio-inspired sensing creates a stream of events that can be much faster and more efficient to process compared to traditional videos for certain applications.
This course teaches students how event cameras work and how to process their data for computer vision tasks. You'll learn why event cameras are useful for tracking fast-moving objects, working in challenging lighting conditions, and building low-power systems. The course covers practical techniques for working with event data, including how to visualize it, convert it to different formats, and apply computer vision algorithms.

We start with a review of basic computer vision concepts, moving to basics of event cameras, and eventually higher level practical applications. Most weeks will include hands-on Python programming exercises where you'll work with real event camera data and implement the techniques discussed in lectures.

Prerequisites

None.

Desired prior knowledge: Image and Video Processing

Recommended reading

None

Introduction to Bio-Informatics

Dept. of Advanced Computing Sciences

BCS3440

Period 2:
27 Oct 2025 **12 Dec 2025**

Credits:
4.0

Coordinator:
R. Cavill

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

This course presents a general introduction to the fundamental methods and techniques of bioinformatics in biomedical and biological research. The objective is that the students will acquire a general understanding of bioinformatics methods at the algorithmic level and will therefore be able to read and understand publications in this field, and – to some extent – apply their knowledge to concrete biological problems. This relates to the major areas of bioinformatics like sequence alignment, phylogenetic analysis, gene finding, and omics data analysis. This course consists of a series of closely related lectures and computer classes, based on relevant case-studies using real data. In the lectures the main theoretical aspects are presented. In the computer practicals, the students work to analyse real data using the techniques they have encountered. By extensively exploring the case study, the students acquire a thorough understanding about the subject.

This is an optional course: Third year students choose three electives per period out of the optional courses during period 1 and 2.

Prerequisites

None.

Desired Prior Knowledge: Procedural Programming, MatLab.

Recommended reading

Introduction to Computational Genomics, A Case Studies Approach, Nello Cristianini, Matthew W. Hahn, Cambridge University Press, 2006, Hardback and Paperback (ISBN-13: 9780521856034 | ISBN-10: 0521856035).

Bachelor's thesis

Dept. of Advanced Computing Sciences

BCS3500

Year:
1 Sep 2025 **19 Jun 2026**

Credits:
18.0

Coordinator:
E. Hortal Quesada

Teaching methods:
Paper(s)

Assessment methods:
Presentation and paper

Full course description

At the end of the Bachelor's study in Data Science and Computing Sciences each individual student has to write a thesis. This thesis has to be designed as a scientific paper of 8 to 10 pages using a standard (LaTeX) design. Students are expected to conduct a pro-active and independent research on their topics. This includes the search and reading of related work. The topics must be discussed with the potential thesis supervisor(s) and a research plan must be submitted to and approved by the Board of Examiners as an initial step. The thesis has to be accompanied by relevant attachments and software. Students will present the thesis in a conference.

This means that a strict submission form will be used.

Prerequisites

In order to start working on the thesis, a student needs to have obtained at least 138 ECTS (among which are 60 credits of the first year, and 40 ECTS of the second year).

For the complete procedure and relevant forms see https://intranet.maastrichtuniversity.nl/en/dke-students/thesis

The History and Philosophy of Computing

Dept. of Advanced Computing Sciences

BCS3410

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
J.W.A.P. WardD.M. Cressman

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

Why do we have the computing technologies that we do? How did these technologies come into being? Are computers capable of intelligence? Can ethics be designed into computer systems, and should it? In this course, students will examine questions like these through the history of computing technologies and the ways in which these technologies have inspired philosophical discussions and debates about computing systems through rigorous thinking.

We will begin with the origins of computing in early twentieth-century business culture through the design of machines to compute ballistics trajectories during World War II to the development of cybernetics, the early days of the internet, networked communication protocols, and the growth of the computing industry. The aim of this history will be to introduce students to the events, ideas, and ambitions that paved the way for today's computing culture. Corresponding with this technological history, we will also examine how these technological changes inspired changing ideas about the place of computing within society, including debates about whether machines can think, the ethics of computing, automation and labour, technological determinism and Moore's Law, and the distinction between communication and information.

At the conclusion of the course, students will have a good understanding of why we have the computing culture that we have today and how this culture came into being. Students will also be able to summarize the philosophical issues that corresponded with this technological history and, in

some instances, link these issues with trajectories in the design of computational systems. Finally, students will learn to look at ethics as something that should be part of the design process of a project and not as something to be "added on" at the final stages.

Prerequisites

None

Recommended reading

None

Operating Systems

Dept. of Advanced Computing Sciences

BCS3420

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
A.I. Iamnitchi

Teaching methods:
Work in subgroups, Project-Centered Learning

Assessment methods:
Written exam, Assignment

Full course description

Description: This course introduces students to the core principles, design strategies, and implementation techniques of modern operating systems. It emphasizes both theoretical knowledge and hands-on practice, bridging the gap between hardware and software. Students will study process management, concurrency, scheduling, memory allocation, and file systems, while engaging in programming projects that simulate real-world system-level challenges. By the end of the course, students will have developed the ability to critically analyze operating system components, implement core mechanisms, and evaluate system performance and reliability.

Knowledge and understanding: By the end of the course, students will understand the role and structure of operating systems as intermediaries between hardware and applications, will be able to explain the principles of process management, describe different memory management techniques such as paging, segmentation, and virtual memory, and assess their trade-offs, and understand the structure and functionality of file systems and I/O subsystems.

Applying knowledge and understanding: Students will apply their knowledge through hands-on labs that utilize operating system APIs for process creation, synchronization, and communication, analyze and solve concurrency problems using synchronization primitives such as semaphores, mutexes, and monitors, and apply performance evaluation tools to assess the behavior of operating system modules and system-level programs.

Making judgements: Students will be required to critically evaluate the efficiency and fairness of scheduling algorithms and memory management strategies; assess trade-offs in operating system design decisions, balancing performance, usability, and security; analyze system failures, deadlocks, or bottlenecks, and propose appropriate solutions; reflect on security considerations in operating system design, especially regarding privacy, access control, and resource sharing.

Communication: Students will develop their communication skills through collaborative problem solving during the labs.

Learning skills: Students will develop skills that enable them to independently explore new operating system concepts and technologies beyond the course content and adapt their understanding of operating systems to emerging technologies, such as virtualization, distributed systems, and cloud computing.

Prerequisites

Desired prior knowledge: Students should be comfortable with basic computer science concepts such as algorithms, data structures, and programming in languages like C, C++ or Java.

Recommended reading

“Operating Systems: Three Easy Pieces”, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. ISBN-10: 198508659X. ISBN-13: 978-1985086593

Theory of Computing

Dept. of Advanced Computing Sciences

BCS3430

Period 4:
26 Jan 2026 **27 Mar 2026**

Credits:
4.0

Coordinator:
G. Stamoulis

Teaching methods:
Project-Centered Learning

Assessment methods:
Written exam

Full course description

The course offers an introduction to the *Theory of Computing* :

- What does “Computation” mean?
- What does it mean something can be computed?
- What quantities can, in principle, be computed? What quantities cannot?
- If something can be computed, *how efficient* can we compute it?

We will give various *models of computation* (deterministic and non-deterministic finite machines) with our ultimate goal to be the definition of the *Turing Machine.* It turns out that all computational models can be defined using a Turing machine as a base. On the 2nd part of the course, we will see that some computational problems are inherently *uncomputable* . This means that, provably and unconditionally, *there cannot be any finite procedure (algorithm) that can solve them* . On the 3rd and final part of the course, we will look at the area of *Computational Complexity Theory* : categorize computational problems according to the resources that are needed in order to solve them or, in other words, what can be computed with limited available resources (such as time or memory)? Typical examples of such complexity classes are P, NP and PSPACE. We will also discuss hierarchy theorems and classes beyond NP.

Prerequisites

None.

Desired Prior Knowledge

- Discrete Mathematics, Design and Analysis of Algorithms

Recommended reading

- I*ntroduction to the Theory of Computation* , Michael Sipser, 3rd Edition.
- *Introduction to Automata Theory, Languages, and Computation* J. E. Hopkroft, R. Motwani, J. D. Ullman 3rd Edition, Pearson Education.
- *Computational Complexity,* Christos Papadimitriou, Addison Wesley 1994.
- *Algorithms* , Jeff Erickson, Independently published 2019
    - Freely available from author's webpage:http://jeffe.cs.illinois.edu/teaching/algorithms
    - Including "Models of Computation" Director's cut (relevant for this course).