Module title		SM Code
Lab Course Computer Science 1		**PIN1**
Module lecturer	**Faculty**	
Prof. Dr. Peter Jüttner	Electrical Engineering and Information Technology	
Module language	**Number of SWS / WSH**	**ETCS credits**
English	2 SWS / WSH	2
Teaching format		
Independent computer-based lab course; supervision upon request		

Semester according to the study plan	
1st semester (Bachelor)	
Attendance/classroom hours	**Additional independent study**
1 hour (submission discussions) up to 23 hours(free allocation)	Preparation and follow-up work: 59 hours (at home or in CIP-Pool). Sufficient preparation and follow-up work for the IN2 submodule is a prerequisite
Type of examination / Requirements for the award of the credit points	
Practical performance assessment	

Teaching content

During the lab course, students will independently solve programming tasks that introduce and deepen their understanding of the various concepts of procedural programming.

Students will implement tasks in C under guidance, with increasingly open-ended questions throughout the semester, requiring independent thinking and thus strengthening their ability to find solutions on their own.

The following topics are covered in particular:

- Basic concepts of procedural programming in C

- Structure of procedural programmes in C: definitions, declarations, statements, expressions, functions
- Elementary data types: declaration, definition, data types, value ranges, internal representation, literal constants, constants, arrays, structured data types
- Operators and expressions: value and side effect, unary and binary operators, operator priority, expressions, families of operators (bitwise, logical, arithmetic, as well as assignment and comparison operators and special operators)
- Statements and control structures: expression statements, multiple statements, branches, loops, functions and function calls
- Distinction between expressions and statements
- Execution model of the C language: functions, memory model, memory management, parameter mechanism, pointers
- The translation process: preprocessor, compiler, linker, multi-part programmes
- Preprocessor: preprocessor symbols, replacement mechanism, conditional compilation, include mechanism, predefined symbols
- Use of the standard library Applications of procedural programming in C
- Applications and algorithm families: finite state machines, sorting methods, random numbers and Monte Carlo algorithms, iterative methods, recursion, simple graphics programming, simple linked lists
- File access: creating, reading and writing files, formatted input and output, line-by-line input and output, binary input and output
- Efficient use of the development environment
- Troubleshooting and use of the debugger

Learning objective: Professional competence

After successfully completing this module, students will be able to independently solve programming problems using procedural programming.

Participants in the course will acquire the following knowledge (10%):

- Basic concepts and terms of procedural programming; knowledge of relevant English technical terms
- Basic language elements of C
- Knowledge of simple standard algorithms

- Basic knowledge of development tools and execution models
- Fundamental insight into the importance of non-functional properties (maintainability, development effort, minimal redundancy in source code, efficient execution, economical use of hardware resources) and possibilities for implementation

Participants in the course will acquire the following skills (2) (60%):

- Implementation of existing algorithms in C
- Understanding foreign implementations
- Independently designing simple algorithms of their own
- Presenting self-developed software solutions and discussing controversial approaches to solutions
- Independently creating procedurally structured software designs and their correct implementation
- Working with development environments
- Independently using debugging tools for troubleshooting

Participants in the course will acquire the following technical and non-technical skills (3) (30%):

- Independent problem analysis and structured problem-solving thinking
- Independent solving of low to medium complexity problems by designing C programmes
- Assessment of the plausibility of programme results
- Testing, debugging and troubleshooting of own and third-party C programmes

Literature

Recommended reading

- Böttcher, A., & Kneißl, F. (2012). *Informatik für Ingenieure* (Third edition). Springer
- Boswell, D., & Foucher, T. (2011). *The Art of Readable Code: Theory in Practice*. O'Reilly
- Wolf, J., & Krooß, R. (2020). *Grundkurs C*. Rheinwerk Computing
- Passig, K., & Jander, J. (2013). *Weniger schlecht programmieren*. O'Reilly
- Kernighan, B. W., & Ritchie, D. M. (1990). *Programming in C. ANSI C*. Hanser

The numbers in brackets indicate the levels to be achieved: (1)-know | (2)-can | (3)-understand and apply