COURSE DESCRIPTION CARD - SYLLABUS

Course name
Programming fundamentals [N1Mech2>PoPr]

Course

Field of study
Mechatronics

Year/Semester
1/1

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
part-time

Requirements
compulsory

Number of hours

Lecture
8

Laboratory classes
16

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

4,00

Coordinators

Lecturers

Prerequisites

Knowledge: A student starting this course should have basic knowledge of computer science, mathematics, computer hardware, computer operation, the Windows operating system, and basic user software. They should also have the ability to obtain information from specified sources and be prepared to collaborate within a team.

Course objective

Providing basic knowledge of computer science, the structure and operation principles of microcomputers, mastering the skills of developing simple algorithms, and the fundamentals of structured programming in C++.

Course-related learning outcomes

Knowledge:
Has structured and theoretically grounded general knowledge of key computer science topics relevant to the field of mechatronics, including programming and the use of IT tools for modeling, simulation, and design.

Skills:
Is able to utilize literary sources, integrate acquired information, assess and interpret it, and draw

conclusions to solve complex and unconventional problems in the field of mechatronics. Can apply appropriately selected methods and tools, including advanced information and communication technologies, and develop simple applications for simulation, analysis, and design of systems relevant to the field of study.

Social competences:
Understands the importance of enhancing professional, personal, and social competencies; is aware that knowledge and skills in the field of mechatronics evolve rapidly. Recognizes the significance of knowledge in solving mechatronics-related problems and acknowledges the necessity of consulting experts when addressing engineering tasks beyond their own competencies.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Assessment of knowledge and skills through a written examination combining test and problem-based questions. Additional points can be earned for active participation during lectures, particularly for: preparing answers to questions and problem tasks assigned by the lecturer, maintaining aesthetic diligence in independently developed assignments, and demonstrating engagement in lectures and laboratory sessions when solving ongoing problem tasks.

Programme content

History of computer science, its areas of application and research. Number systems, fixed-point and floating-point number representations, information encoding, fundamentals of digital circuits, computer system architecture, buses, general characteristics of processors, RAM, and ROM. Operating systems, computer networks, network computing, and issues of computer network security. The Internet, intranet. Algorithms and data structures. Selected algorithms for solving analytical problems in mathematics and physics, as well as sorting algorithms. Programming languages. The C++ programming language. Structured programming. Programming in the C++ Builder/Visual C++ environment. Laboratory: fundamentals of programming in C++ (syntax, development of simple algorithms and programs).

Course topics

Lecture:
I. Programming languages. Compilation, interpretation. Low-level and high-level languages. Arithmetic operations, logical operators, standard functions, conditional instructions, sequential instructions
II. Functions. Defining functions. Returning a result through a function. Passing arguments to a function by value and by reference. Arrays, enumeration types, vectors.
III. Recursion. Definition and principles of recursion. The base case and the recursive case. Pointers. Defining pointers. Dynamic memory allocation. Passing arguments to a function via a pointer.
IV. Test.
Laboratories:
I. Introduction to the Visual Studio Development Environment
• Overview of the computer lab regulations
• Overview of occupational health and safety (OHS) regulations and course completion requirements
• Introduction and configuration of the Visual Studio environment, Console
• Preprocessor directives - e.g., #using, #pragma, #define
• Overview of C++ syntax, variable types, variable declaration, constants - structured programming
II. Arithmetic Operations, Logical Operators, Standard Functions, Conditional Statements
• Basic arithmetic and logical operators - programs illustrating the operation of operators and standard functions
• Control statements: if, if...else, switch, break, continue, goto
• Programs testing the operation of the above statements using operators
III. Programming Instruction Sequences
• Control statements: while, do..while, for, foreach (Range-based for)
• Selected programs testing sequential instructions (Calculating series sum, summing n odd numbers, finding the maximum value of a sequence of numbers entered from the keyboard, selecting numbers divisible by 3 from a sequence, searching for a perfect number, searching for a prime number)

IV. Functions
• Defining functions
• Returning results from a function
• Passing arguments to functions by value and by reference
• Developing sample functions (greatest common divisor, least common multiple, decimal to binary conversion, factorial calculation, average value calculation, etc.)
V. Arrays, Enumerations, Vectors
• Defining arrays, single and multidimensional static arrays
• Array handling
• Defining vectors, single and multidimensional vectors
• Vector handling
VI. Pointers
• Defining pointers
• Null and nullptr expressions
• Dynamic arrays
• Dynamic memory allocation
• Passing arguments to functions via pointers
VII. Structures and Unions
• Defining structures
• Defining unions
• sizeof expression
• Operations on structures and unions
VIII. Exam

Teaching methods

Applied Teaching Methods: a) Lecture with a multimedia presentation (including drawings, photos, animations, sound, and videos), supplemented with examples presented on the board. b) Interactive lecture with questions directed to the group or specific students. c) Student activity during classes is considered in the final assessment. d) Theory is presented in close connection with practice and the students' current knowledge.
Laboratory: Demonstrations and independent execution of programming (computational) tasks.

Bibliography

Basic:
1. Jerzy Grębosz - "Opus magnum C++11. Programowanie w języku C++" (2017)
2. Jerzy Grębosz - "Opus magnum C++. Misja w nadprzestrzeń C++14/17" (2023)
3. Stephen Prata - "Język C++. Szkoła programowania. Wydanie VI" (2012)
4. Bjarne Stroustrup - "Programowanie. Teoria i praktyka z wykorzystaniem C++" (2011)

Additional:
1. Scott Meyers - "Skuteczny nowoczesny C++" (2016)
2. Bruce Eckel - "Thinking in C++. Edycja polska. Tom 1" (2002)
3. Bruce Eckel, Chuck Allison - "Thinking in C++. Edycja polska. Tom 2" (2003)

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,00
Classes requiring direct contact with the teacher	24	1,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	76	3,00