COURSE DESCRIPTION CARD - SYLLABUS

Course name
Informatics [S1EiT1>INF]

Course

Field of study
Electronics and Telecommunications

Year/Semester
1/2

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

6,00

Coordinators

dr inż. Michał Sybis
michal.sybis@put.poznan.pl

Lecturers

Prerequisites

Basic knowledge of mathematical logic and combinatorics. Ability to formulate simple algorithms. Can obtain information from literature and other sources in Polish or English; is able to integrate the obtained information, interpret it and draw conclusions. Knows the limitations of his own knowledge and skills, and understands the need for further learning.

Course objective

Acquainting the basics of software engineering. The course introduces further issues of both the practice of object-oriented programming of computers in C++ as well as the design of data structures and algorithms and the analysis of their computational complexity.

Course-related learning outcomes

Knowledge:
1. Basic theoretical and practical knowledge in the field of programming in C and C ++, with particular emphasis on the design of correctly constructed programs, the principles of object-oriented software construction, the use of templates, the design of complex programs and the use of libraries.
2. Knowledge of basic algorithms (sorting, searching data sets, greedy methods, trial and error methods,

selected numerical algorithms) and data structures (containers, one-way and two-way lists, binary search trees, balanced trees, graphs and methods of searching them, dendrites) used in the programmer"s daily practice.
3. Review knowledge of the methods used in designing complex IT instruments and object-oriented designing of specialized data structures.

Skills:
1. When designing software, the student is able to analyze the problem from the point of view of algorithmic proceedings using the criteria of computational complexity, program speed, scalability of the solutions used, and the adequacy of the methods adopted.
2. When designing software, the student is able to decompose the problem properly, select adequate data structures, extract the hierarchy of objects, identify relations between objects and their representatives in the program.
3. The student is able to critically analyze the available library software in terms of its use in the implemented project and propose the principles of cooperation in the configuration of a collective programmer.

Social competences:
1. Understanding the need for a wider popularization of knowledge in the field of modern IT techniques.
2. Awareness of the possibilities and limitations of modern computer science, while being open to the possibility of applications in new areas of everyday life, economy, technology and science.
3. The ability to form one"s own opinions on the currently used and available technologies and solutions in the design of modern information systems.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Scope of the first semester (out of two):
The knowledge and skills acquired during the lectures are verified during the exam. The exam has a written or oral form (it is allowed to conduct part of the exam in a written way and part in an oral way). It consists of 10-15 open questions, which do not have to be scored equally. The passing threshold for the written exam is 50% - 60% of the possible points.
The skills acquired during the laboratory classes are assessed based on: a short test at the beginning of the classes, an assessment of students' work in class and at home, and a final test at the end of the semester (or two tests in the middle and at the end of the semester, respectively). Weights of each criteria: admission fee up to 25%, evaluation of the student's work in class and at home up to 50% and test (or the sum of results from both tests) with a weight of at least 50%. The passing threshold is 50% of the points available. In addition, the student has the opportunity to obtain additional points by completing additional tasks that may increase the final grade.
Scope of the second semester (out of two):
The knowledge and skills acquired during the lectures are verified during the exam. The exam has a written or oral form (it is allowed to conduct part of the exam in a written way and part in an oral way). It consists of 10-15 open questions, which do not have to be scored equally. The passing threshold for the written exam is 50% of the possible points.
The skills acquired during the laboratory classes are assessed based on: a short test at the beginning of the classes (the so-called pass), an assessment of students' work in class and at home, and a final test at the end of the semester (or two tests in the middle and at the end of the semester, respectively). Weights of each criteria: admission fee up to 25%, evaluation of the student's work in class and at home up to 50% and test (or the sum of results from both tests) with a weight of at least 50%. The passing threshold is 50% - 60% of the possible points. In addition, the student has the opportunity to obtain additional points by completing additional tasks that may increase the final grade.

Programme content

The scope of the lecture in the first semester (out of two) includes program structure in C++, basic data types, data conversion, representation of binary fixed and floating point numbers (including the IEEE 754 standard), operators and expressions, bitwise operations, control statements, arrays, functions, argument passing, function patterns, function overloading, recursive algorithms, sorting algorithms, fast sorting algorithms, computational complexity, binary search, hash table as a data structure.
The scope of the lecture in the second semester (out of two) includes classes and class objects,

constructors, destructors, operator overloading, pointers and dynamic memory allocation, class patterns, basic data structures from the STL database (vectors, lists, stacks, queues, trees, graphs), the idea of iterators, inheritance, polymorphism, object-oriented programming, modern software engineering, reverse Polish notation, binary trees, binary tree searching, AVL trees, graph theory problems, graph searching, Euler's cycle, Hamilton's cycle, Prim's and Kruskal's algorithms, finding algorithms shortest paths (Alg. Dijkstra, Alg. Bellman-Ford, Alg. Floyd).
The scope of laboratory classes in the first semester (out of two) includes familiarization with the idea of programming and the programming language, creating variables and operations on variables, creating arrays and operations on arrays, creating functions, overloading functions, recursive functions, passing arguments to functions, sorting methods, binary search, hash tables.
The scope of laboratory classes in the second semester (out of two) includes the basics of object-oriented programming in C++: creating classes, methods, objects, operator overloading, inheritance, and polymorphism. Creating dynamic data structures: one-way list, two-way list, binary search tree. Performing operations on data structures, sorting, searching, adding and removing elements, etc. Basics of using the STL library.

Course topics

none

Teaching methods

Lecture: multimedia presentation illustrated with examples given on the board. Laboratories: practical exercises - implementation of tasks given by the teacher.

Bibliography

Basic
1. Jerzy Grębosz, Symfonia C++ : programowanie w języku C++ orientowane obiektowo. T. 1/2/3, 2000
2. Jerzy Grębosz, Pasja C++ : szablony, pojemniki i obsługa sytuacji wyjątkowych w języku C++. T. 1/2, 2004
3. Jerzy Grębosz, Opus Magnum C++11 : programowanie w języku C++. T. 1/2/3, 2018
4. Bjarne Stroustrup, Programowanie : teoria i praktyka z wykorzystaniem C++ ; przekład Łukasz Piwko, 2020
5. Bjarne Stroustrup, C++ : podróż po języku dla zaawansowanych; tłumaczenie: Łukasz Piwko, 2019
6. Bjarne Stroustrup, The C++ programming language, 2018

Additional
1. Slobodan Dmitrović, Modern C++ for Absolute Beginners: A Friendly Introduction to the C++ Programming Language and C++11 to C++23 Standards, 2023
2. Ivor Horton i Peter Van Weert, Beginning C++20: From Novice to Professional, 2020
3. Bjarne Stroustrup, A Tour of C++, 2022
4. Stephen Prata, Język C++. Szkoła programowania, 2022

Breakdown of average student's workload

	Hours	ECTS
Total workload	300	12,00
Classes requiring direct contact with the teacher	150	6,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	150	6,00