COURSE DESCRIPTION CARD - SYLLABUS

Course name
Informatics [S1EiT1>INF2]

Course

Field of study Electronics and Telecommunications	Year/Semester 2/3
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 (e.g. online) 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, understands the need for further learning.

Course objective

Acquainting with the basics of software engineering. The subject introduces further issues of both the practice of object-oriented programming of computers in C ++ and the design of data structures and algorithms as well as 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:

Learning outcomes presented above are verified as follows:
The scope of the first semester (out of two):
The knowledge and skills acquired during the lectures are verified during the exam. The exam is in writing form. It consists of 11-13 open-ended questions that do not need to be scored equally. The pass mark for the written exam is 50% of the points available. It is also possible to verify students' knowledge on the basis of an individual or team project and its defense.
The skills acquired during the laboratory classes are assessed on the basis of: a short test at the beginning of classes (the so-called entrance test), evaluation of students" work during classes, and a final test taking place in 15 classes (or two tests in the middle and at the end of the course, respectively). Weights of individual criteria: entrance test maximum 25%, student work grade in class maximum 50% and test (sum of results from both tests) weighing at least 50%. The pass mark is 50% of the possible points.
The scope of the second semester (out of two):
The knowledge and skills acquired during the lectures are verified during the exam. The exam is in writing. It consists of 11-13 open-ended questions that do not need to be scored equally. The pass mark for the written exam is 50% of the points available. It is also possible to verify students' knowledge on the basis of an individual or team project and its defense.
The skills acquired during the laboratory classes are assessed on the basis of: a short test at the beginning of classes (so-called entrance test), evaluation of students" work during classes, and a final test taking place in 15 classes (or two tests in the middle and at the end of the course, respectively). Weights of individual criteria: entrance test maximum 25%, student work grade in class maximum 50% and test (sum of results from both tests) weighing at least 50%. The pass mark is 50% - 60% of the possible points. In addition, the student has the opportunity to obtain additional points through tickets and additional tasks that may increase the final grade.

Programme content

The lecture for the first semester includes the basics of programming in C++, covering program structure, data types, control statements, arrays, functions, and the fundamentals of computation theory and algorithms, such as sorting and binary search. The second semester focuses on object-oriented programming, introducing classes, objects, inheritance, polymorphism, and advanced data structures and graph algorithms. Laboratory sessions in the first semester concentrate on practical aspects of basic programming, such as operations on variables, arrays, functions, and basic sorting and searching algorithms. In the second semester, the labs develop skills in object-oriented programming and operations on dynamic data structures, utilizing the STL library.

Course topics

The scope of the lecture in the first semester (out of two) includes: program structure in C ++, basic data types, data conversion, complementary representation of binary numbers, operators and expressions, bitwise operations, control statements, arrays, functions, argument passing, function patterns, overloading functions, calculation theory, recursive and greedy algorithms, sorting algorithms, fast sorting algorithms, computational complexity, binary search, hash array as data structure.
The scope of the lecture in the second semester (out of two) includes: classes and class objects, constructor, destructor, operator reloading, 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, class patterns, object-oriented programming, modern software engineering, inverse Polish notation, binary trees, searching binary trees, AVL trees, graph theory issues, graph searching, Euler, Hamilton cycle, Prim, Kruskal algorithm , algorithms for finding the shortest paths (Dijkstra alg., Bellman-Ford alg., Floyd alg.).
The scope of laboratory classes in the first semester (out of two) includes: familiarization with the idea of programming and programming language, creating variables and operations on variables, creating arrays and operations on arrays, creating functions, overloading functions, recursive function, 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 overload, inheritance, 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.

Teaching methods

Lecture: multimedia presentation, illustrated with examples given on the board.
Laboratories: practical exercises - carrying out the 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. Bruce Eckel, Thinking in C++. Polish edition.
Additional
1. Stephen Prata, Język C++.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Wprowadzenie do algorytmów, WNT, Warszawa, 2004.

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