COURSE DESCRIPTION CARD - SYLLABUS

Course name
Algorithms and data structures 1 [S1Teleinf1>AiSD1]

Course

Field of study
Teleinformatics

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 (e.g. online)
0

Tutorials
0

Projects/seminars
0

Number of credit points

5,00

Coordinators

dr inż. Filip Idzikowski
filip.idzikowski@put.poznan.pl

prof. dr hab. inż. Jerzy Tyszer
jerzy.tyszer@put.poznan.pl

Lecturers

Prerequisites

The student should have basic knowledge of discrete mathematics, combinatorics and probability theory. They should have the ability to perform calculations using mathematical apparatus in the field of mathematical analysis and probability, and to obtain information from the indicated sources.

Course objective

The course aims at introducing students to the area of algorithms and data structures. Furthermore, it presents methodologies and techniques of the object oriented programming using C++, providing a fairly complete introduction to the language.

Course-related learning outcomes

Knowledge
Students know basic principles and rules used to design effective object oriented programs and data structures. They also know details regarding various algorithms used to handle numerical and discrete

math problems. They also learn how to design data structures with the help of templates and standard libraries.
Skills
A student can design an algorithm using, as guiding criteria, its time and memory complexity. He/she is also capable of coding proposed algorithms by deploying languages such as C++. A student understands the concept of object-oriented programming and impact of various data structures on time and memory efficiency of software applications.
Social competences
A student appreciates the practical significance of the object-oriented programming paradigm. Is aware of limitations of various algorithms. Is open for new applications of software engineering in technology, science, and social (daily) life. Can express his/her own opinions with respect to currently used solutions and methods as far as design of contemporary software systems is concerned.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

2h-long written exam comprising assignments that cover the content of lectures. Laboratory classes are evaluated based on several written tests and a few small projects.

Programme content

Software engineering, the main classes of algorithms. Program structure in C++, fundamental data types and instructions. Binary representation of integers, bitwise operators. Computational complexity, problems P and NP, Turing machine. Functions, stack, passing arguments to functions, overloading of functions, templates, lambda expressions. Recursion. Basic sorting algorithms. Binary heap and heapsort. Quicksort and merge sort. Binary search. Hashing. Binary representation of floating point numbers. Errors in floating point computations. Solving equations. Finding min and max. Integration rules. Gauss elimination. Lexicographic order, permutations, transpositions, Steinhaus-Johnson-Trotter algorithm, k-element sets.

Course topics

Lecture: software engineering, the main classes of algorithms. Program structure in C++, fundamental data types. Arithmetic and Boolean expressions, relations, casting, instruction blocks. Conditional instructions, selection, loops, tables. Binary representation of integers, two's complement coding, bitwise operators, Eratosthenes sieve. Computational complexity, problems P and NP, Turing machine. Functions, stack, passing arguments to functions, overloading of functions, templates, lambda expressions. Recursion, Euclid's algorithm. Basic sorting algorithms – selection sort, insertion sort, bubble sort, Shell's method. Binary heap and heapsort. Quicksort and merge sort. Binary search. Hashing. Selection of a hashing function. Collisions. Binary representation of floating point numbers. Errors in floating point computations. Solving equations. Bisection, Newton-Raphson method. Finding min and max. Integration rules. Gauss elimination. Lexicographic order, permutations, transpositions, Steinhaus-Johnson-Trotter algorithm, k-element sets, k-element random sets.

Labs: Getting familiar with the lab environment. Basic operations on files. Simple C++ programs: input/output streaming, seeking max and min values, computations on matrices. Basic control flow instructions. Functions – passing arguments, templates, examples. Recursion – common algorithms. Simple sorting methods (insertion sort, selection sort, bubble sort). Heapsort, Quicksort. Binary search. Hashing and handling collisions. Numerical methods: bisection, Newton-Raphson method. Generation of simple combinatorial objects.

Teaching methods

Lectures: a multimedia presentation.

Laboratory classes: students solve various problems provided by a teacher, write programs, compile them, debug a code, and evaluate programs on benchmark tests.

Bibliography

1. R. Sedgewick, Algorytmy w C++, Oficyna Wydawnicza READ ME, Łódź, 1999
2. N. Wirth, Algorytmy + struktury danych = programy, WNT, Warszawa, 1980.

3. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Wprowadzenie do algorytmów, WNT, Warszawa, 2004
4. E.W. Dijkstra, Umiejętność programowania, WNT, Warszawa, 1985.
5. J. Grębosz, Symfonia C++, Oficyna Kallimach, Kraków 2008.
6. W. Lipski, Kombinatoryka dla programistów, WNT, Warszawa, 1982.

Breakdown of average student's workload

	Hours	ECTS
Total workload	120	5,00
Classes requiring direct contact with the teacher	64	3,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	56	2,00