COURSE DESCRIPTION CARD - SYLLABUS

Course name
Algorithms and data structures 2 [S1Teleinf1>AiSD2]

Course

Field of study
Teleinformatics

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
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. Should be able to perform calculations using a 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:
The student has basic theoretical and practical knowledge of programming in C and C++, with an emphasis on designing well-formed programs, designing complex programs, and using library software. The student

also has knowledge of basic algorithms and data structures used in everyday programming practice.

Skills:
When designing software, the student can analyze a problem from an algorithmic perspective, applying criteria of computational complexity, scalability of the solutions used, and adequacy of the methods adopted. They can also critically analyze available library software for its application in a project and propose principles of collaboration within a collaborative programming configuration.

Social Competencies:
The student is aware of the possibilities and limitations of modern computer science while being open to possible applications in new areas of everyday life, economics, technology, and science.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Knowledge acquired during the lecture is assessed through a written exam consisting of several problem-solving exercises covering the lecture content and/or a multiple-choice test of approximately 15 questions. Skills acquired during laboratory exercises are assessed through two written tests covering tasks completed during the course. Furthermore, skills acquired during the course are continuously assessed through project exercises and oral presentations. Participation in the classes is also assessed.

Programme content

Object oriented programming. C++ classes. Constructors and destructors. Friend functions. Operator overloading. Inheritance. Hierarchy of classes. Templates. Pointers. Dynamic creation of objects. Polymorphism. Virtual functions. Copy constructor. Containers. A linked list. Iterators. Ordered lists, bidirectional lists, cyclic lists, queues. Reverse Polish Notation. Binary trees. Binary search trees. AVL trees. Insertion to and deletion from AVL trees. Graphs. Depth-first search, Euler cycles, finding Euler paths. Hamiltonian cycle. Travelling salesman problem, simulated annealing. Breadth- first search. Spanning trees. Dijkstra's algorithm.

Course topics

Lecture: Object oriented programming. C++ classes. Constructors and destructors. Complex numbers - implementation. Constructor as a converter. Friend functions. Operator overloading. Inheritance. Hierarchy of classes. Templates. Pointers, operators new and delete. Dynamic creation of objects. Pointers to complex objects. Polymorphism. Virtual functions. Abstract classes. Copy constructor. Containers. A linked list, basic operations on lists. Iterators and their applications. Ordered lists, bidirectional lists, cyclic lists, queues. Reverse Polish Notation. Binary trees, basic operations, traversal methods. Binary search trees – insertion and deletion. AVL trees – local balancing, rotations of nodes. Insertion to and deletion from AVL trees. Graphs. Depth-first search, Euler cycles, Euler graphs, finding Euler paths. Hamiltonian cycle. Travelling salesman problem, simulated annealing. Breadth- first search. Spanning trees. Prim's, Kruskal's, and Boruvka's algorithms. Dijkstra's algorithm, the Euclidean metric.

Labs: forming simple classes. Creation of objects. Constructors and destructors. Operator overloading. Friend functions. Inheritance and class templates. Polymorphism and virtual functions. Containers – simple examples. Containers for vectors. Forming linked lists. Binary trees and traversal methods: in- order, pre-order and post-order. Working with binary search trees – insertion and deletion. AVL trees. Building graphs. Prim's and Kruskal's algorithms. Dijkstra's algorithm.

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