COURSE DESCRIPTION CARD - SYLLABUS

Course name
Structured and object-oriented programming [N1AiR2>PSiO]

Course

Field of study
Automatic Control and Robotics

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
part-time

Requirements
compulsory

Number of hours

Lecture
20

Laboratory classes
20

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

5,00

Coordinators

dr inż. Piotr Kaczmarek
piotr.kaczmarek@put.poznan.pl

Lecturers

Prerequisites

A student starting this subject should have basic knowledge of computer hardware and its operation, and of the courses of semester I: Fundamentals of Computer Science and Information Technology.

Course objective

Purpose of the course: 1. Acquainting with the methodology and principles of structured and object-oriented programming using the C ++ programming language in the scope extended to that presented in semester I and elements of Python. 2. Acquainting with dynamic data structures and their implementation in C ++ and Python. Developing practical skills of adequate use of structures depending on the requirements 3. Ability to implement and adapt standard algorithms to solve a variety of problems, and issues related to computational complexity and optimization 4. Knowledge of basic application design patterns and an example of their use

Course-related learning outcomes

Knowledge:
The graduate has an orderly knowledge of selected algorithms and data structures as well as methodology and techniques of procedural and object-oriented programming. The graduate knows and understands

basic processes occurring in the software development cycle.

Skills:
The graduate can construct an algorithm for a simple engineering task and implement, test and run it in a selected development environment on a PC for selected operating systems.

Social competences:
The graduate is ready to critically evaluate his or her knowledge. The graduate understands the need for and knows the possibilities of continuous learning - improving professional, personal and social competences, the graduate is able to inspire and organize the learning process of others.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Lecture: written exam covering the lecture
Laboratory: testing practical skills in the field of algorithms and data structures
object-oriented programming in C++, and the ability to use C++ STL. The grade is the result of the test grade, the grade of class work and homework assignments
and evaluation of the final project.

Programme content

The lecture program includes the following topics:
- Dynamic data structures (array, list, tree, hash table, stack, graph): construction, implementation in various programming languages (C++, Python), as well as performance and applications.
- Algorithms: algorithm complexity, recursive and iterative approaches, sorting and searching, algorithms for tree structures, graph algorithms.
- Design patterns (including Model-View-Controller, Model-View, Singleton, Decorator, Strategy, Observer, Adapter).
- STL C++11,14 (containers and algorithms, predicates, regular expressions), generic programming (templates).

The laboratory program includes:
-Exercises on the implementation of algorithms based on data structures in C++.
-Object-oriented programming.
-Use of the STL standard library in C++, including algorithms, containers, and changes introduced in new versions of the library.
-Practical use of design patterns during the final project implementation.

Course topics

The program of the lecture and laboratory classes covers the following issues:
- dynamic data structures (array, list, tree, hash table, stack, graph) structure, implementation in C ++, as well as performance and applications,
- algorithms: algorithm complexity, recursive and iterated approach, sorting and searching, algorithms for tree structures, graph algorithms
- design patterns (including Model Control View, Model View, Singleton, Dekorator, Strategy, Observator, Adapter)
- STL C ++ 11,14 (containers and algorithms, predicates, regular expressions), generic programming (templates).

Teaching methods

1. Lecture: multimedia presentation, illustrated with examples given on the board, and with programs created during the classes.
2. Laboratory exercises: practical exercise on C++, supported by didactic materials placed on the e-learning platform

Bibliography

Basic:

1. Opus Magnum C++11 : programowanie w języku C++. T. 1-3 / Jerzy Grębosz. Wydawnictwo Helion, cop. 2018.
2. materiały dydaktyczne udostępnione dla zajęć laboratoryjnych i wykładu: https://moodle.put.poznan.pl
3. Brad Miller and David Ranum “Problem Solving with Algorithms and Data Structures using Python” Luther College 2018 (dostępna online)

Additional:
1. B. Eckel, Thinking In C++, Edycja polska, Wydawnictwo Helion
2. Podstawy programowania C i C++ - skrypt/P. Kaczmarek, D. Belter. Wydawnictwo Politechniki Poznańskiej 2011

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,00
Classes requiring direct contact with the teacher	42	2,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	83	3,00