COURSE DESCRIPTION CARD - SYLLABUS

Course name
Introduction to programming

Course

Field of study	Year/Semester
Automatic Control and Robotics	1 / 1
Area of study (specialization)	Profile of study
Control and Robotics Systems	general academic
Level of study	Course offered in
First-cycle studies	Polish
Form of study	Requirements
part-time	compulsory

Number of hours

Lecture	Laboratory classes	Other (e.g. online)
18	18	0
Tutorials	Projects/seminars	
0	0	

Number of credit points

5

Lecturers

Responsible for the course/lecturer:	Responsible for the course/lecturer:
dr inż. Rafał Kapela	dr inż. Jakub Kołota
email: Rafal.Kapela@put.poznan.pl	email: Jakub.Kolota@put.poznan.pl
tel. 61 6652184	tel. 61 6652751
Wydział Automatyki, Robotyki i Elektrotechniki	Wydział Automatyki, Robotyki i Elektrotechniki
ul. Piotrowo 3, 60-965 Poznań	ul. Piotrowo 3, 60-965 Poznań

Prerequisites

Skills: Should have the ability to efficiently use a PC and external devices, as well as the ability to obtain information from specified sources. He should also have a basic knowledge of the basics of automation and control theory. He should have the ability to actively participate in organized lectures for a large group of people, being aware of the need to expand theoretical and practical knowledge and to constantly update acquired knowledge due to dynamic technological and systemic changes in modern technology.

Social Competences: He should also understand the need to broaden his competences and be ready to cooperate as part of a team implementing e.g. a joint project.

Course objective

The purpose of the education module:

1. Acquainting with the methodology and principles of object oriented programming using the C ++ programming language.
2. Developing students ability to solve problems in the area of modeling and implementation of information systems. Students learn to simulate and analyze the operation of object-oriented IT programs, and plan and document the IT work done.
3. Developing programming skills for students. Creating awareness of the need for a professional approach to technical issues, meticulous familiarization with the documentation of UML-type IT systems. The student learns to set goals and define priorities leading to the implementation of the task through object-oriented implementation of the code.

Course-related learning outcomes

Knowledge

1. has advanced knowledge of selected algorithms and data structures as well as methodologies and techniques of procedural and object-oriented programming; - [KW_8]

2. has structured knowledge of computer architectures, computer systems and networks, and operating systems including real-time operating systems; - [KW_9]

Skills

1. is able to critically use information from literature, databases and other sources in Polish and a foreign language; - [KU_1]

Social competences

1. understands the need and knows the possibilities of continuous training - raising professional, personal and social competences, is able to inspire and organize the learning process of others; - [K_K1]

2. is aware of the responsibility for own work and readiness to comply with the rules of teamwork and taking responsibility for jointly implemented tasks; can manage a small team, set goals and set priorities for achieving them; is ready to perform responsible professional roles; - [K_K3]

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:

a) in terms of lectures: based on answers to questions about the material discussed in previous lectures,

b) in the scope of laboratories / exercises: based on the assessment of the current progress of tasks,

Summative rating:

a) in the scope of lectures, verification of assumed learning outcomes is carried out by:

i. assessment of knowledge and skills demonstrated during the problem-type written exam consisting of 5 questions out of 40 questions presented on the general list of questions, previously provided to students.
Grading rules:
5.0 - above 90% of exam points (W); average of grades from lab exercises above 4.75 (L)
4.5 - 80% -90% of exam points (W); average of grades from lab exercises 4.25-4.75 (L)
4.0 - 70% -80% of exam points (W); average of grades from lab exercises 3.75-4.25 (L)
3.5 - 60% -70% of exam points (W); average of grades from lab exercises 3.25-3.75 (L)
3.0 - 50% -60% of exam points (W); average of grades from lab exercises 2.75-3.25 (L)
2.0 - less than 50% of exam points (W); average grades from lab exercises below 2.75 (L)
ii. discussion of passing results,
b) in the scope of laboratories / exercises, verification of assumed learning outcomes is carried out by:
a. assessment of student's preparation for individual laboratory sessions (entrance test)
b. assessment of the laboratory exercise carried out (report)

Programme content

The lecture program includes the following issues:

During the semester, the lecturer comprehensively discusses C ++ semantics in object-oriented terms. Each lecture is dedicated to a different issue and presents solutions in the form of ready implementations. Students receive lecture materials in the form of files containing ready source files along with the code discussed during a given lecture. The content of the lecture is supplemented with two lectures on UML modeling, during which the student gets acquainted with the method of creating documentation and UML notation (class and object diagram, activity diagram, sequence diagram, use case diagram, etc.). Laboratory exercises were prepared in the form of pdf files , which are tasks for self-realization during classes. For this purpose, the student uses the student's account on the e-learning portal of the Department of Computer Engineering (http://dydaktyka.cce.put.poznan.pl/moodle) and downloads a document with the content of the task. During the class, the teacher briefly recalls the program content related to the given exercise (discussed earlier by the teacher during the lecture). Further laboratories cover the following issues of object oriented programming:

- C ++ language - repetition of the first semester and leveling of the basic level of the laboratory group
- Class and object - quantifiers of the private and public classes of the class
- Constructor (copy constructor), destructor, static class components (static modifier)
- Indicators, dynamic arrays (moving around and dynamically allocating / releasing it)
- Inheritance (protected quantifier, multilevel inheritance issues) and mechanism of friend functions
- Virtual methods and the issue of polymorphism
- Class abstraction
- Overloading functions and operators
- Type casting and conversion
- STL, list container (list) and array container (vector)

Part of the above-mentioned program content is implemented in the student's own work.

Teaching methods

1. lecture: multimedia presentation, presentation illustrated with examples given on the board, as well as multimedia shows and demonstrations using, among others CodeBlocks or gcc/g++ programs

2. practical exercises, implementation of tasks and algorithms defined in studies entrusted to the student, discussion on the complexity and optimization of the code during classes (all materials developed electronically and embedded on the e-learning platform)

Bibliography

Basic
1.M. Dadaj, Programowane zorientowane obiektowo. Wyd. Helion 2005.

2. B. Stroustrup, Język C++, wydanie V, WNT, Warszawa 2000

3. Jerzy Grębosz, Symfonia C ++ Standard, Editions 2000, Kraków 2005

4. Zbigniew Koza, Język C++. Pierwsze starcie, Helion, Gliwice, 2008

Additional
1. B. Eckel, Thinking In C++, Edycja polska, Wydawnictwo Helion.

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5
Classes requiring direct contact with the teacher	36	2
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation)[1]	89	3

1 delete or add other activities as appropriate