STUDY MODULE DESCRIPTION FORM

Name of the module/subject **Languages and paradigms of programming**	Code **1010331541010334960**

Field of study **Information Engineering**	Profile of study (general academic, practical) **(brak)**	Year /Semester **2 / 4**
Elective path/specialty **-**	Subject offered in: **Polish**	Course (compulsory, elective) **obligatory**

Cycle of study: **First-cycle studies**	Form of study (full-time,part-time) **full-time**

No. of hours								No. of credits
Lecture:	**30**	Classes:	**-**	Laboratory:	**30**	Project/seminars:	**-**	**4**

Status of the course in the study program (Basic, major, other) **(brak)**	(university-wide, from another field) **(brak)**

Education areas and fields of science and art	ECTS distribution (number and %**)**
technical sciences	**4 100%**
Technical sciences	**4 100%**

Responsible for subject / lecturer:

dr inż. Grażyna Brzykcy
email: grazyna.brzykcy@put.poznan.pl
tel. 616653714
Faculty of Electrical Engineering
ul. Piotrowo 3A 60-965 Poznań

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	Student has basic knowledge of mathematics, especially in such fields as algebra, analysis and logic, basic knowledge of program constructs, implementation of algorithms, formal languages and programming platforms.
2	**Skills**	Student is able to use basic techniques to create algorithms, to analyze their complexity, and to use software platforms and environments for simple programs encoding, running and testing.
3	**Social competencies**	Student understands the importance of stringent accomplishment of a given project with proper notation standards.

Assumptions and objectives of the course:

Presentation of declarative programming styles and rules of choosing the adequate style and language to a class of problems. Development of declarative programming skills in functional and logic programming environments.

Study outcomes and reference to the educational results for a field of study

Knowledge:

1. Student has organized and theoretically founded knowledge of creation, implementation and applicability of recursive data structures. - [[K_W04]]

2. Student has organized and theoretically founded knowledge of computation models and basic declarative program constructions. - [[K_W05]]

3. Student is familiarized with state of the art and current trends in programming paradigms. - [[K_W19]]

Skills:

1. . Student is able to create engineer work documentation and declaratively present the work result. - [[K_U03]]

2. Student can use techniques of logic and functional programming to create algorithms. - [[K_U09]]

3. Student is able to use declarative software platforms and environments for simple programs encoding, running and testing. - [[K_U10]]

Social competencies:

1. Student understands and is aware of the importance of issues related to computer engineer activity. Student understands the responsibility for his engineering decisions. - [[K_K02]]

2. Student understands the importance of stringent accomplishment of a given project with proper notation standards, proper language. Student understands the importance of keeping deadlines. - [[K_K07]]

Assessment methods of study outcomes

Lecture

Written test based on lecture (basic concepts and techniques used in declarative programming).

Laboratory

Students? marks are based on continuous assessment of their programming activity and results of two written tests (creation of simple programs).

Course description

Lectures

Logic as programming language (procedural aspect of SLD-resolution). Data structures and procedures in Prolog. Recursive data structures and recursive programs. Functional programming: data types, functions, overview of languages and environments. Current trends in declarative programming. Some non-classical programming techniques: evolutionary computation, constraint-based programming, artificial neural networks.

Teaching methods:

- presentation of the theory with frequent references to relevant practical examples of software implementations,
- lecture with multimedia presentation and examples dawn on a blackboard,
- students being asked questions during the lectures in order to provoke discussions.

Course update 2017:

- Erlang introduced as functional programming language,
- artificial neural networks as another programming paradigm.

Laboratory

Creation of algorithms and their implementation in declarative programming languages: logic programming language Prolog, and functional programming language Erlang.

Teaching methods:

- presentation of short generic programs,
- students define individual solutions of simple problems.

Laboratory update 2017:

- new programming environment Erlang.

Basic bibliography:

1. Haber F.: Learn you someERLANG for great good! A beginner's guide (on-line learnyousomeerlang.com), 2017.
2. Kowalski R., Logic for problem solving, North-Holland, 1979.
3. Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, 3rd edition, Springer-Verlag, Berlin, 1996.
4. Nilsen U., Małuszyński J.: Logic, Programming, and PROLOG, John Wiley & Sons, 2000.
5. Van Roy P., Haridi S., Concepts, Techniques, and Models of Computer Programming, The MIT Press, 2004.

Additional bibliography:

1. Armstrong J.: Programming Erlang. The Pragmatic Programmers, 2013.
2. Cesarini F., Thompson S.: Erlang Programming. O'Reilly Media, 2009.
3. Mozart Consortium, The Mozart programming system, http://www.mozart-oz.org, 2006.

Result of average student's workload

Activity	Time (working hours)
1. Lecture	30
2. Laboratory	30
3. Preparation to laboratory and tests	40
4. Sterling L., Shapiro E.: The Art of Prolog. Advanced Programming Techniques, MIT Press, 1986.	0

Student’s workload

Source of workload	hours	ECTS
Total workload	100	4

Contact hours	60	2
Practical activities	70	3