**SYLLABUS**

**regarding the qualification cycle FROM 2024TO 2025**

1. Basic Course/Module Information

|  |  |
| --- | --- |
| Course/Module title | Languages and Programming Paradigms |
| Course/Module code \* |  |
| Faculty (name of the unit offering the field of study) | College of Natural Sciences |
| Name of the unit running the course | Institute of Computer Science |
| Field of study | Computer Science & Computer Science and Econometrics |
| Qualification level | First degree |
| Profile | Academic |
| Study mode | Full-time |
| Year and semester of studies | Year 3, semester 6 |
| Course type | Major |
| Language of instruction | English |
| Coordinator | Krzysztof Balicki, PhD |
| Course instructor | Krzysztof Balicki, PhD |

\* - as agreed at the faculty

1.1. Learning format – number of hours and ECTS credits

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Semester  (n0.) | Lectures | Classes | Colloquia | Lab classes | Seminars | Practical classes | Internships | others | **ECTS credits** |
| 6 |  |  |  | 30 |  |  |  |  | 5 |

1.2. Course delivery methods

- conducted in a traditional way

- involving distance education methods and techniques

1.3. Course/Module assessment (exam, pass with a grade, pass without a grade)

- pass with a grade

2. Prerequisites

|  |
| --- |
| The basics of programming in C language, Algorithms and data structures, Object-oriented programming |

3. Objectives, Learning Outcomes, Course Content, and Instructional Methods

3.1. Course/Module objectives

|  |  |
| --- | --- |
| O1 | The course covers the four most significant programming paradigms: imperative, object-oriented, functional and logic programming, and related programming languages. |
| O2 | This course is designed to give participants a broader view of programming. It shows standard features and differences between the languages typical for these paradigms and the methods of creating and compiling programs written in these languages. |
| O3 | An essential feature of these classes is the large number of practical exercises that enable students to put into practice the learned paradigms. |

3.2. Course/Module Learning Outcomes (to be completed by the coordinator)

|  |  |  |
| --- | --- | --- |
| Learning Outcome | The description of the learning outcome  defined for the course/module | Relation to the degree programme outcomes |
| LO\_01 | The student knows the rules of formulation and algorithmization of tasks and the basic notation of algorithms. | K\_W04 |
| LO\_02 | The student knows the basic programming paradigms and at least one language representing each of the learned paradigms to the extent that allows him to write simple application programs. | K\_W07 |
| LO\_03 | The student can specify and analyse (including tracking) an algorithm compliant with the specification and write it in the selected programming language. | K\_U11 |
| LO\_04 | The student can write simple application programs in at least one language of functional programming and programming in logic. | K\_U12 |

**3.3. Course content (to be completed by the coordinator)**

1. Lectures

|  |
| --- |
| Content outline |
| 1. Overview of the most significant programming paradigms. |
| 2. Function programming in Haskell. |
| 3. Declarative programming in Prolog. |
| 4. Imperative programming – a reminder of the main points. |
| 5. Object-oriented programming – a reminder of main points. |
| 6. Imperative paradigm – review, and reminder of main points. |
| 7. Use of different programming paradigms to solve algorithm problems. |
| 8. Other programming paradigms - summary lecture. |

1. Classes, laboratories, seminars, practical classes

|  |
| --- |
| Content outline |
| 1. Programming environment. Compilers and interpreters. |
| 2. Functional programming in the Haskell language. |
| 3. Declarative programming in the Prolog language. |
| 4. Imperative programming in C - revision of selected topics. |
| 5. Object-oriented programming in Java - revision of selected topics. |
| 6. Use of various programming paradigms to solve algorithmic problems. |

3.4. Methods of Instruction

Lecture: a lecture supported by a multimedia presentation

Laboratory: solving exercises, implementation projects

4. Assessment techniques and criteria

4.1 Methods of evaluating learning outcomes

|  |  |  |
| --- | --- | --- |
| Learning outcome | Methods of assessment of learning outcomes (e.g. test, oral exam, written exam, project, report, observation during classes) | Learning format (lectures, classes,…) |
| LO-01 | observation during classes | lab |
| LO-o2 | test | lab |
| LO-o3 | test | lab |
| LO-04 | test | lab |

4.2 Course assessment criteria

|  |
| --- |
| Lab  To complete the lab, you must pass the Haskell and the Prolog test. The final grade is the average of marks from these tests. You must earn at least half of the maximum number of points to complete the test. Test grades are proportional to the number of points scored. Activity in the labs is also considered, as it may lower or increase the final score by half a degree.  Lectures  The learning outcome of the lectures is verified during the laboratory classes with the help of tests and observation. |

5. Total student workload needed to achieve the intended learning outcomes

– number of hours and ECTS credits

|  |  |
| --- | --- |
| Activity | Number of hours |
| Scheduled course contact hours | 30 |
| Other contact hours involving the teacher (consultation hours, examinations) | 5 |
| Non-contact hours - student's own work (preparation for classes or examinations, projects, etc.) | 90 |
| Total number of hours | 125 |
| Total number of ECTS credits | 5 |

\* One ECTS point corresponds to 25-30 hours of total student workload

6. Internships related to the course/module

|  |  |
| --- | --- |
| Number of hours | - |
| Internship regulations and procedures | - |

7. Instructional materials

|  |
| --- |
| Compulsory literature:   1. P. Hudak, J. Peterson, J. Fasel: A Gentle Introduction to Haskell <https://www.haskell.org/tutorial/index.html> 2. W.F.Clocksin, C.S.Mellish, Prolog. Programowanie, Wydawnictwo Helion, 2003 |
| Complementary literature:   1. Joeren Fokker: Functional Programming. Department of Computer Science, Utrecht University 1995 (pdf file available on the Internet) 2. Hal Daume III, et. al.: Yet Another Haskell Tutorial. 2004 (plik pdf dostępny w Internecie) 3. J. R. Fischer: Prolog tutorial, http://www.csupomona.edu/~jrfisher/www/prolog\_tutorial/contents.html 4. Dave Stuart Robertson: Quick Prolog, http://www.dai.ed.ac.uk/groups/ssp/bookpages/quickprolog/quickprolog.html 5. Patrick Blackburn, Johan Bos and Kristina Striegnitz: Learn Prolog Now!, http://www.learnprolognow.org 6. Brian W. Kernighan, Dennis Ritchie: Język ANSI C. WNT, Warszawa 2003 7. Bruce Eckel: Thinking in Java. Edycja polska, Wydanie 4, Helion, Gliwice, 2006 8. Marcin Lis: Praktyczny kurs Java. Helion, Gliwice, 2007 |

Approved by the Head of the Department or an authorised person