COURSE DESCRIPTION CARD - SYLLABUS

Course name
Advanced Programming [S2Bioinf2>ZPR]

Course

Field of study
Bioinformatics

Year/Semester
1/1

Area of study (specialization)
–

Profile of study
general academic

Level of study
second-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other (e.g. online)
0

Tutorials
0

Projects/seminars
0

Number of credit points

4,00

Coordinators

dr inż. Marcin Radom
marcin.radom@put.poznan.pl

prof. dr hab. inż. Marta Kasprzak
marta.kasprzak@put.poznan.pl

Lecturers

dr inż. Marcin Radom
marcin.radom@put.poznan.pl

prof. dr hab. inż. Marta Kasprzak
marta.kasprzak@put.poznan.pl

Prerequisites

Students commencing the Bioinformatics second degree program should have achieved the educational results of the first degree program, as defined in the resolution of the Senate of PUT - these results are presented on the department's website cat.put.poznan.pl. In particular, students starting this course should have the knowledge and skills in the following subjects from the Bioinformatics first degree: Algorithms and Data Structures, Foundations of Programming in C, Object-Oriented Programming.

Course objective

1. To provide students the knowledge of the .NET programming platform. 2. To introduce students to the principles of constructing polynomial metaheuristic algorithms and exponential exact algorithms. 3. To teach students how to design and implement advanced algorithms for complex combinatorial problems derived from computational biology, using the .NET platform.

Course-related learning outcomes

Knowledge:
As a result of the course, the student knows:
1. methods, techniques and tools used in the process of solving complex bioinformatics tasks, mainly of engineering nature, including advanced methods of combinatorial optimization,
2. principles of creating and testing complex software for bioinformatics problems,
3. principles of planning research in the development of new algorithmic solutions for computationally hard bioinformatics problems.

Skills:
As a result of the course, the student will be able to:
1. plan and perform advanced computational experiments and interpret their results,
2. apply advanced techniques and computer tools to solve biological problems and evaluate their usefulness,
3. design and implement a complex software project, including a generator of test instances and a metaheuristic, according to a given specification, taking into account non-technical aspects,
4. prepare a comprehensive written paper in Polish presenting results of his/her own research,
5. apply a systems approach to solving bioinformatics tasks, taking into account non-technical aspects.

Social competences:
Successful completion of the course means that the student is ready to:
1. identify priorities to accomplish a task defined by self or others,
2. take responsibility for decisions made and comply with the principles of professional ethics.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:
a) In the field of lectures, verification of the assumed educational effects is realized by answers to questions concerning the material discussed during lectures.
b) In the field of laboratory classes, verification of the assumed educational effects is realized by evaluation of skills related to the implementation of laboratory exercises.
Summing up assessment:
a) In the field of lectures, verification of the assumed educational effects is realized by assessment of knowledge and skills related to the content provided during lectures in a form of one collective colloquium. It has the form of a multiple-choice quiz with eight questions, implemented through the e-kursy platform. The maximum number of points is 8, the passing threshold is 4 points (>=5 points - grade 3.5, >= 5.5 points - grade 4.0, >= 6.5 points - grade 4.5, >=7.5 points - grade 5.0).
b) In the field of laboratory classes, verification of the assumed educational effects is realized through the assessment of knowledge and skills related to the content taught during lectures and laboratory classes, made on the basis of the final project presented by the student, the report and the "defense" of the project.

Programme content

The programme content covered within the course includes practical aspects of creating advanced software to solve a complex bioinformatics problem, as well as the theory enabling the development of such software from scratch with the use of appropriate approaches.

Course topics

Lectures start with refreshing knowledge of C# language elements: the .NET programming environment, its architecture, basic .NET features and other C# related constructs. These include: program structure, data types and operations on them, program modularization - defining and calling functions, classes, object-oriented programming in C#: encapsulation, inheritance, polymorphism, interfaces, files and serialization. Collection classes and functions provided by them. Parametric collection classes (generics). Delegation: defining type, creating and using delegation object (delegation arrays, multiple delegations, anonymous delegations, delegation patterns). Lambda expressions. Events: defining an event associated with a delegation, registering functions in an event, reporting an event. Exception handling. Concurrency: creating concurrent threads and managing these threads. Concurrent execution of tasks, concurrent execution of calculations. Windows Forms library. Used namespaces, the initial form and its

properties. Adding controls, defining their properties and defining event handling functions. Mouse and keyboard controls, menus, status bar, toolbars. Basic controls: buttons, text boxes, drop-down lists, etc. Creating graphs. Dialog boxes: standard and custom dialogs.
A series of lectures presenting ways of designing algorithms begins with a reminder of basic concepts of combinatorial optimization and an explanation of the terms heuristics and metaheuristics. Genetic algorithms: general principles, parameters, representation of an individual, fitness function, generation of an initial population, selection, crossover, mutation, intensification and diversification strategies. Tabu search method: general principles, parameters, generation of an initial solution, definition of a move and a neighborhood, tabu list, aspiration criterion, memory types, intensification and diversification strategies. Heuristics: constructive vs. improving, greedy heuristics, beam search, problem decomposition, local search, hyperheuristics. Other metaheuristics: genealogy and classification, GRASP, evolutionary programming, genetic programming, scatter search, ant colony optimization, particle swarm optimization, hybrid heuristics. Exact algorithms: application constraints, branch & bound, branch & cut, dynamic programming.
Laboratory classes are conducted in the form of 15 two-hour classes held in a computer laboratory. The first class is to acquaint students with rules of using the laboratory and the software. The program of laboratory classes includes the following issues: preparation of applications in which elements are coded in C# language, developing modularized programs using collection classes, interfaces, events and delegations, developing programs using disk files. All programs are using Windows Forms architecture for creating windows-based applications.
Within the laboratory classes a final project is realized. The project includes design, implementation and testing of one metaheuristic - genetic algorithm or tabu search method of polynomial time complexity - for one chosen bioinformatics problem from given ones. The metaheuristic is to be specialized, i.e., configured to properties of a particular problem in order to optimize the quality of obtained solutions. In addition, the project includes implementation of a procedure that generates non-trivial instances in a random manner, with parameters giving instances of varying difficulty, especially large ones. This procedure will enable in-depth testing of capabilities and limitations of the metaheuristic, and results of an extensive computational experiment will be included in the report.

Teaching methods

1. Lecture: multimedia presentation, presentation illustrated with examples given if necessary.
2. Laboratory exercises: preparation of programs in .NET environment allowing to get to know particular technologies present in this environment; discussion of methods used for the completion of the final project.

Bibliography

Basic:
1. Pro C# 7.0 With .NET and .NET Core”, 8th Edition, Andrew Troelsen , Philip Japikse, wyd. Apress, 2017
2. Pro C# 8 with .NET 3. Foundational Principles and Practices in Programming”, 9th Edition, Andrew Troelsen, Phip Japikse, wyd. Apress, 2020
3. Metaheuristics: From Design to Implementation, El-Ghazali Talbi, Wiley, Hoboken, 2009
4. Jak to rozwiązać czyli nowoczesna heurystyka, Zbigniew Michalewicz, David B. Fogel, WNT, Warszawa, 2006

Additional:
1. Programowanie w C#, wydanie VI, Ian Griffiths, Matthew Adams, Jessy Liberty, O’Reilly 2010, Helion 2012
2. Algorytmy genetyczne i ich zastosowania, David E. Goldberg, WNT, Warszawa, 1995
3. Tabu Search, Fred Glover, Manuel Laguna, Kluwer Academic Publishers, Boston, 1997
4. Złożoność obliczeniowa problemów kombinatorycznych, Jacek Błażewicz, WNT, Warszawa, 1988

Breakdown of average student's workload

	Hours	ECTS
Total workload	120	4,00
Classes requiring direct contact with the teacher	60	2,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	60	2,00