COURSE DESCRIPTION CARD - SYLLABUS

Course name
Foundations of Programming in C

Course

Field of study
Bioinformatics

Year/Semester
1/1

Area of study (specialization)
-

Profile of study
general academic

Level of study
First-cycle studies

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other (e.g. online)

Tutorials

Projects/seminars

Number of credit points

6

Lecturers

Responsible for the course/lecturer:
Marcin Radom, Ph.D.
Faculty of Computing and Telecommunications

Responsible for the course/lecturer:

Prerequisites

Student should have a starting knowledge about computer architecture, basic computer science terminology and be proficient in english langage on a basic level.

Course objective

1. Basic and intermediate knowledge about C programming language.

2. Basics of programming in general and structure-oriented programming in particular.

3. Giving students knowledge and skills to solve algorithmic problems.

4. Development of a skill how to divide a problem into subproblems which make creating a solving algorithm possible.

5. Learning about Integrated Development Environments (IDE) for programming languages on the example of Code::Blocks.

Course-related learning outcomes

Knowledge

As a result of the course, the student knows:

1. elements of the algorithms and the C language (e.g., loops, conditional instructions, other commands that make up the canon of the language), dynamic data structures (lists, queues, trees) and the elements from computational complexity theory.

2. principles of structured programming in ANSI C and related languages.

Skills

As a result of the course the student is able to:

1. obtain information from the literature and from other appropriately selected sources (e.g., websites about programming languages), also in English,

2. design and develop computer software according to the given specifications, using appropriate methods, techniques and tools,

3. acquire knowledge and improve qualifications required due to frequently changing programming styles and models.

Social competences

Successful completion of the course means that the student:

1. understands the need for lifelong learning and improving their competences due to the continuous development of programming languages, the emergence of new coding standards, etc.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:

a) in the scope of lectures, verification of the assumed educational effects is realized by the answers to the questions concerning the material discussed during previous lectures

b) in the field of laboratory, the verification of the assumed educational effects is realized through the evaluation and "defense" of the laboratory exercises realized by the student and the evaluation of knowledge and skills related to the implementation of the laboratory tasks through two written tests per semester.

Summative assessment:

a) in the scope of lectures, the verification of the assumed educational effects is realized by the assessment of knowledge and skills demonstrated at the written examination in the form of a multiple-choice test and a few open questions. The exam consists of 10-20 questions with a total value of 20-40 points distributed depending on the difficulty of the question. Students receive a passing grade after scoring a minimum of half the points.

b) in the scope of laboratory classes the verification of the assumed learning outcomes is realized by:

- During the realization of laboratory classes students have to write up to two colloquia, concerning programming issues discussed during the lecture. Additionally, they have to write a credit program on their own according to the specifications discussed during the classes.

- The student is given a positive mark for the laboratory if he/she successfully completes both tests and correctly implements the credit program.

The activity during the classes is rewarded with additional points, in particular for the effectiveness of the application of the acquired knowledge while writing the assigned program in a manner exceeding the minimum specified in the specification.

Programme content

The lecture program covers issues of broadly understood structured programming on the basis of the ANSI C language. During the first lecture the students are additionally introduced to the most basic IT issues required in programming (such as: memory organization on the lowest levels as bits, bytes, words, differences between an algorithm and a program, introduction to popular programming environments like Eclipse, Visual Studio, etc.). From the second meeting begins a series of lectures devoted to the basic elements of ANSI C such as loops, conditional instructions, variables, functions/procedures/methods, structures and their objects, etc. After covering the basics necessary for programming, there follows a sequence of lectures devoted to the more theoretical aspects of algorithmics. These lectures cover topics such as graphs, data structures like lists, queues or binary trees, different sorting algorithms and their comparison from the point of view of time and memory complexity.

Laboratory exercises are conducted in the form of 15 two-hour classes held in a computer laboratory. The first class is devoted to acquaint students with the rules of using the laboratory and passing the exercises. The program of the laboratory classes includes the following:

- practicing and consolidating knowledge from lectures on various elements of the ANSI C language,

- writing simple programs on the basis of the practiced knowledge,

- development of a larger program as part of the laboratory classes to illustrate such concepts as code transparency, consistent writing style, division of the program into different functional units, etc.

- exercises on more difficult issues necessary to master before writing a credit program independently.

Teaching methods

1. Lecture: presentations in PowerPoint/PDF forms and additional examples in necessary.

2. Laboratories: solving problems, practical problems, working in a team.

Bibliography

Basic

1. B.W.Kernighan, D.M. Ritchie, Język ANSI C, Wydawnictwa Naukowo-Techniczne, seria Klasyka Informatyki

2. C.L. Tondo, S.E. Gimpel, Język ANSI C – ćwiczenia i rozwiązania, Wydawnictwa Naukowo-Techniczne, seria Klasyka Informatyki

Additional

1. http://pl.wikibooks.org/wiki/C

Breakdown of average student's workload

	Hours	ECTS
Total workload	150	6,0
Classes requiring direct contact with the teacher	60	3,0
Student's own work (preparation for laboratory classes, preparation for tests and the exam, project preparation) [1]	90	3,0

[1] delete or add other activities as appropriate