Fundamentals of programming - course description

General information

Course name	Fundamentals of programming
Course ID	13.2-WF-FizP-FP-S17
Faculty	Faculty of Physics and Astronomy
Field of study	Physics
Education profile	academic
Level of studies	First-cycle studies leading to Bachelor's degree
Beginning semester	winter term 2018/2019

Course information

Semester	2
ECTS credits to win	5
Course type	obligatory
Teaching language	english
Author of syllabus	• prof. dr hab. Mirosław Dudek

Classes forms

The class form	Hours per semester (full-time)	Hours per week (full-time)	Hours per semester (part-time)	Hours per week (part-time)	Form of assignment
Lecture	15	1	-	-	Exam
Laboratory	60	4	-	-	Credit with grade

Aim of the course

The aim of the course is learning the basics of programming and the ability to use knowledge to solve a variety of problems with special emphasis on the sciences. This approach to programming requires an understanding of not only the syntax of a programming language, but also the basics of algorithms, coding standards, the ability to work with documentation and analysis of a complex problem and reduce the number of elementary problems. Programming is the essential foundation for understanding the following subjects such as computer numerical methods and computer simulations.

Prerequisites

It is assumed that students have a basic knowledge of working in a Linux environment. Prerequisites are a subset of the material carried out on the first computer lab.

Scope

1. Safety at work, rules of the computer lab.

2. Introduction to programming

- The concept of the algorithm, the strategy of "divide and conquer"
- Programming concept, cycle "analysis - code-execution",
- The types of errors
- Source code and machine code, compilation, linking
- History on programming languages

3. Introduction to C language
- variable types, excess and underflow errors,
- declarations and definitions of variables, projections of types
- local and global variables
- output to the screen
- mathematical operators
- C language syntax
- naming conventions and source code formatting
- compilation and linking

4. Conditions
- logical operators
- conditional operator
- conditional statement

5. Loops
- loop while
- loop do while
- loop for

6. Arrays
- one-dimensional arrays

- multidimensional arrays

7. Functions
- concept of function
- parameters passed to functions, default parameters, return values
- range of validity of variables
- references
- recursion
- work with arrays

8. Indicators and variables part.2.
- indicators
- references
- static and const prefixes
- dynamic memory allocation
- indicator arrays
- parameters passed during program launch

9. Structures
- concept of structure
- structures as data types
- use of structures

10. IO operations
- binary and text data streams
- standard input and output streams
- reading data from files
- saving to files

-

Teaching methods

Lecture:
Conventional lectures, discussion, workshops (currently testing the code fragments), brainstorming

Laboratory: exercises, project method, group work, exchange ideas, brainstorming, presentation, working with documents, self-knowledge acquisition

Learning outcomes and methods of theirs verification

Outcome description	Outcome symbols	Methods of verification	The class form
Student can independently find and use the tools and information to solve a given problem.		• a quiz	• Laboratory
Student is aware of Open Source software which is a professional alternative to commercial software. Student is aware of the rate of change in the IT industry and thus the need for continuing to improve own competence. Students can create and present a report on the project.		• a discussion	• Laboratory
Student can compile and run the program, and interpret the results of problems in physics (or similar), as well as to verify the correctness of the action on the basis of the acquired knowledge in a particular field.		• a quiz	• Laboratory
Student knows the data types, control statements, functions, and can work arrays and IO streams. Can apply the acquired knowledge and available tools to provide solution to the problem (in particular in the field of physics and related fields) in source code form.		• a quiz • an exam - oral, descriptive, test and other	• Lecture • Laboratory
Students can work in a group, feels responsible for the tasks assigned to it, is open to new concepts and ideas.		• a discussion	• Laboratory
Student knows the safety rules in the computer lab.		• a quiz	• Laboratory
Student is able to define and explain the problem stated for elementary problems and provide methods (algorithms) of optimal solution of the problem.		• a quiz • an exam - oral, descriptive, test and other	• Lecture • Laboratory

Assignment conditions

Lecture:
The practical test consists in solving a given problem (the drawn from the list of problems). The final rating is the analysis of the problem, provide algorithms for solving the problem, the source code and the assessment and verification of the results.

Laboratories:
Score: average grades achieved during the laboratories of activity and short tests advances in science (50% of the final mark) and the assessment of the semester project (50% of the final mark). To pass the semester project is its preparation and commitment within the prescribed period of the project report and its presentation.

Before taking the exam the student must obtain a pass from the laboratory.

Score: weighted average rating of the exam (60%) and exercise (40%).

Recommended reading

[1] Allen Downey, *Think Python. How to Think Like a Computer Scientist*, 2013, Green Tea Press Needham, Massachusetts.

[2] Stephen Prata „Język C, Szkoła programowania”, Helion 2016

Further reading

[1] Internet

Notes

Lectures should be in the room with internet access. Computer labs should be done in groups to allow individual work with each student's computer and not with more than 12 people.

Modified by dr hab. Piotr Lubiński, prof. UZ (last modification: 01-08-2018 14:36)