COURSE DESCRIPTION CARD - SYLLABUS

Course name
Introduction to Computing [S1Inf1>WdIang]

Course

Field of study
Computing

Year/Semester
1/1

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
English

Form of study
full-time

Requirements
elective

Number of hours

Lecture
30

Laboratory classes
16

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

4,00

Coordinators

prof. dr hab. inż. Jerzy Nawrocki
jerzy.nawrocki@put.poznan.pl

Lecturers

Prerequisites

When starting this course, the student should have the knowledge and skills specified in the regulation of the Minister of National Education of 30 January 2018 on the core curriculum of general education for general secondary schools, technical schools, and vocational schools of the second degree (https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20180000467/O/D20180467.pdf): * mathematics: extended scope, * computer science: basic scope.

Course objective

The course aims to help students understand what computer science is by introducing them to the basic areas of computer science and showing the relationships between them. Moreover, the course supports students in acquiring basic programming skills.

Course-related learning outcomes

Knowledge:
1 has a structured and theoretically founded general knowledge in the field of key issues of computer science, and detailed knowledge in the field of selected issues in this discipline of science
2. has knowledge of important directions of development and the most important achievements of

computer science and other related scientific disciplines, in particular electronics, telecommunications as well as automatic control and robotics
3. has basic knowledge about the life cycle of IT systems, both hardware and software, and in particular about the key processes taking place in them
4. has knowledge of ethical codes related to IT, is aware of the dangers related to electronic crime, and understands the specificity of mission-critical systems
5. has basic knowledge of patents, the copyright and related rights act and the act on the protection of personal data and technology transfer, in particular with regard to IT solutions .

Skills:
1. can properly use information and communication techniques, applicable at various stages of the implementation of IT projects
2. can see in the process of formulating and solving IT tasks also non-IT aspects, in particular social, legal and economic issues
3.can organise, cooperate and work in a group, assuming different roles in it, and can properly define priorities for the implementation of a task set by himself or others

Social competences:
1. understands that in computer science knowledge and skills very quickly become obsolete
2. is aware of the importance of knowledge in solving engineering problems and knows examples and understands the causes of malfunctioning information systems that have led to serious financial and social losses or to serious health conditions or even to death

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

The verification of the assumed learning outcomes is carried out by:
- ongoing assessment in the form of short quizzes;
- mid-semester test;
- assessment of teamwork;
- exam.

Programme content

The course includes the following content:
Imperative programming
Digital circuits
Computers and assembly language
Subroutines
Text processing
Object-oriented programming
Numerical methods
Computational complexity
Databases and machine learning
Concurrent computing
Computer networks and cryptography
Software engineering
Professionalism in computer science

Course topics

Lectures and associated labs cover the following topics:
* Comparison of C and Python: simple programs, declaring and processing simple variables, reading and printing variables, conditional statements, loop statements, arrays, records, modularization concepts, and functions;
* Basic digital circuits: Boolean algebra, logic gates, adders, decoders and multiplexers, flip-flops, registers, digital biochemical circuits;
* Building a simple computer: von Neumann architecture, microprocessors vs. microcontrollers, Turing machine;
* NASM assembly language: integer instructions, simple programs, hexadecimal system, integers,

representation of negative numbers, jump instructions and their implementation;
* Subroutines in C and Python: passing parameters, function as a parameter, implementation of subroutines in assembly language;
* Text processing in C and Python: string representation, text processing, standard input/output library elements, regular expressions, AWK;
* Object-oriented programming basics in C++ and Python: records vs. classes, inheritance, dynamic memory allocation, lists;
* Numerical methods: representation of real numbers - IEEE 754-1985 standard, numerical stability, Newton's method and square root, Maclaurin series, and trigonometric functions;
* Computational complexity: big O notation, types of computational complexity, the complexity of heap sorting, brute force and its complexity, optimization and decision problems, polynomial transformation of problems, halting problem;
* Databases: file processing, relational data model, SQL language, projection, selection, join operation, data maintenance, patterns, query nesting, grouping, database manipulation from Python;
* Machine learning: main approaches to machine learning, supervised learning concept, information entropy, ID3 method;
* Parallel programming: evolution of operating systems, processes vs. threads, time sharing, computational interference, POSIX standard, pthreads library, thread management, access synchronization, semaphores, producer-consumer problem, readers-writers problem, deadlock;
* Computer networks: layered architecture of computer networks, stack of Internet protocols, packets, TCP/IP protocol and TCP sockets, HTTP protocol, HTML and JavaScript language, building web services;
* Software engineering: problem analysis, functional requirements and use cases, selected UML diagrams, user interface design, non-functional requirements, software architecture, implementation and testing;
* IT Professionalism: principles of effectiveness, win-win principle, synergy, open source and software licenses, patents.

Teaching methods

Lecture: multimedia presentation and quizzes
Laboratory exercises: solving tasks, programming, discussion

Bibliography

Basic:
1.Język ANSI C, B. W. Kernighan, D. M. Ritchie, Wydawnictwa Naukowo-Techniczne, dowolne wydanie
2. Układy cyfrowe, B. Wilkinson, WKiŁ, Warszawa, 2000
3. Wprowadzenie do przetwarzania tekstów w języku AWK, J. Nawrocki, W. Complak, ProDialog 2, 23-46, Poznań, 1994.
4. Sieci komputerowe, J.F. Kurose, K.W. Ross, Helion, 2006

Additional:
1. 7 habits of highly effective people, S. Covey, Simon & Schuster, 1999
2. Język C. Szkoła programowania, Wydanie VI, Prata S., Helion, 2016

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,00
Classes requiring direct contact with the teacher	48	2,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	52	2,00