SYLLABUS

Name: ***Fundamentals of Computer Programming (InfAAu>SI1FoCP19)***
Name in Polish:
Name in English: ***Fundamentals of Computer Programming***

Information on course:

Course offered by department: Faculty of Automatic Control, Electronics and Computer Science
Course for department: Silesian University of Technology

Default type of course examination report:
EGZ
Language:
English
Course homepage:
https://platforma2.polsl.pl/rau2/course/view.php?id=912
Short description:
The course is an introduction to structural programming in a high level language. The course has two parts: a lecture and a laboratory. The course covers structural programming in C++ (organisation of a program, functions, lambdas, streams, files, pointers, STL data structures). In the laboratory class students practise these topics, good programming practices, application design, data structure choice.
Description:
ECTS:4
Classes:
* total workload: 120 h (30 contact hours + 60 h student work)
* lecture 30h
* laboratory 30h

Student's work: preparation for the laboratories, implementation of the programming task.

The course develops practical skills.
The course develops engineering competences.
Przedmiot kształtuje umiejętności praktyczne.
Przedmiot kształtuje kompetencje inżynierskie.

covered topics:
* program structure and organisation
* types, variables, operators
* enum types
* arrays
* loops (while, do-while, for, range for)
* functions (declarations, definitions, argument passing, array passing, recursion)
* variable scope, automatic variables
* structures, unions
* STL containers STL: array, vector, string, map, unordered_map, pair, tuple, list, forward_list, queue, priority_queue
* input / output operations in C++
* pointers (raw and intelligent), heap variables, allocation and desallocation of memory in C++
* source and header files
* lambda functions
* namespaces
* STL algorithms
* input / output operations in C
* strings in C
* allocation and desallocation of memory in C
* preprocessor

lecture and laboratory topics:

W01
* program structure
* types, variables, operators (including <=>)

* conditional statement (if + ?:)
* logical operators

W02
* loops (while, do-while, for(;;))
* function (simple function -- idea)

W03
* switch statement
* fixed-size arrays
* two-dimensional arrays
* enumeration type (enum, class enum)

W04

* local blocks and variables, variable scopes, automatic variables
* functions (declaring, methods of passing arguments, arrays, recursion, inline, constexpr, consteval, default function parameters, overloading)

W05
* main parameters
* input/output operations (std::cout, std::cerr, std::cin, file streams, string stream)

W06
* stream manipulators
* stream error flags
* header files
* std::array

W07
* std::vector (create, push_back, [], size, capacity, initializer list), for (:))
* std::string
* structures
* unions

W08

* std::pair (make_pair, first, second)

* associative arrays (map)

W09

* pointers
* pointers to variable, constant
* function output parameters (via pointer)
* arrays (pointer approach)

W10

* function pointers
* typedef, using
* iterators
* lambda functions

W11
* lambda functions (completion)
* STL containers: (vector,) deque, (string, map), list, forward_list, set, priority_queue, stack, (pair,) tuple

W12
* STL containers (continued): list, forward_list, set, priority_queue, stack, (pair,) tuple
* std::format

W13

* memory allocation, memory leaks, smart pointers, array allocation (single and multidimensional)

W14
* smart pointers continued
* binary files
* C strings (idea, strlen, strcpy, strcmp)

W15
* C I/O (printf, scanf, files)
* C heap memory allocation (malloc, calloc, realloc, free)
* preprocessor (macro definitions, conditional compilation, X macro)

Sylabus obowiązuje od roku akademickiego 2025/26 a jego zawartość nie podlega zmianom w trakcie trwania semestru.

Bibliography:

1. Eckel, B., Allison, C. „Thinking in C++: Practical Programming"
2. Bjarne Stroustrup: „The C++ Programming Language"
3. Robert C. Martin: „Clean Code"
4. Standard C++ Library reference, http://www.cplusplus.com/reference/

Learning outcomes:

knowledge: knows and understands
EU_W1 The student knows the language constructs for structured programming (K1A_W09)
EU_W2. The student understands what a function is in programming and its role in code modularity. (K1A_W11, K1A_U07)
EU_W3. The student knows the properties of data structures (K1A_W12)

skills: can

EU_U1. The student can select data structures appropriate to the problem. (K1A_U21)
EU_U2. The student can implement a program in a high-level language that solves a given problem (K1A_U23)
EU_U3. The student can apply good programming practices (K1A_U23)
EU_U4. The student can verify the correctness of the implementation (K1A_U15)
EU_U5. The student can prepare technical documentation for the created program (K1A_U03)

social skills: is ready for
EU_K1. The student is ready to search, critically evaluate information and formulate coherent and justified conclusions. (K1A_K03)

Assessment methods and assessment criteria:

The exam has two parts: (A) a practical part (programming task) and (B) an oral part.

A. practical part – programming task
1. The practical part is mandatory.
2. The task topic is assigned by the instructor.
3. The task is performed by one person.
4. The full source code of the program must be uploaded into a git repository.
5. The program code must be commented in the doxygen standard.
6. The task documentation generated in doxygen must be placed on the subject page on the distance learning platform.
7. The programming task consists of assessed components.
8. The lab instructor determines the deadline for submitting each component.
9. Personal presentation of each component of the project is a necessary condition for determining the degree of fulfillment of the learning outcomes.
10. The instructor may verify the authorship of the program in a manner he deems appropriate.
11. The grade for the programming task is the arithmetic mean of the components, where each component must be graded positively.
12. In the event of failure in the practical part of the exam, taking the next practical part requires implementing a new programming task assigned by the laboratory instructor. This means that all components of the programming task must be graded again.

B. Oral part
1. The oral part is optional.
2. A necessary condition for taking the oral part is a positive grade for the practical part.
3. Passing the practical part exempts you from the oral part with a grade of 3.0.
4. The oral part is graded on a scale of 2 to 5, where positive grades are grades from 3 (sufficient) to 5 (very good).

The final grade for the subject is the arithmetic mean of the grades from the practical and oral parts, both of which must be positive.

The syllabus is valid from academic year 2025/26 and its contents cannot be changed during the semester.

Course credits in various terms:

<without a specific program>

Type of credits	Number	First term	Last term
European Credit Transfer System (ECTS)	4	2020/2021-Z	