FACULTY OF ELECTRONICS

SUBJECT CARD

Name of subject in Polish:	**Wstęp do programowania**
Name of subject in English:	**Introduction to Programming**
Main field of study (if applicable):	**Electronic and Computer Engineering**
Specialization (if applicable):	**……..**
Profile:	**academic**
Level and form of studies:	**1 st level/ full-time**
Kind of subject:	**obligatory**
Subject code:	**ECEA00002**
Group of courses:	**YES**

	Lecture	Classes	Laboratory	Project	Seminar
Number of hours of organized classes in University (ZZU)	30		45		
Number of hours of total student workload (CNPS)	120		120		
Form of crediting	crediting with grade		crediting with grade		
For group of courses mark (X) final course	**X**				
Number of ECTS points	**8**				
including number of ECTS points for practical (P) classes			4		
including number of ECTS points corresponding to classes that require direct participation of lecturers and other academics (BU)	2		4		

*delete as applicable

PREREQUISITES RELATING TO KNOWLEDGE, SKILLS AND OTHER COMPETENCES

\

SUBJECT OBJECTIVES

C1 Acquisition of basic knowledge on computer algorithms, how they are presented and analyzed. Getting familiar with standard algorithms processing large amounts of data, i.e.: searching, aggregating and sorting.

C2 Learning the basic programming constructs which are common to most of algorithmic languages: types, variables, conditional branching, looping, functions with arguments, recursion, arrays, lists, files. Getting Acquainted with selected forms of dynamic and complex data structures: list, stack, queue and tree.

C3 Acquiring the ability of the structural and procedural programming in C or C++, and using the integrated development environments to improve the processes of editing, compiling and testing multi-file programming projects.

C4 Update and development of the knowledge in the area of information technologies and improving the skills in its use in engineering work.

SUBJECT EDUCATIONAL EFFECTS	
relating to knowledge:	
PEU_W01	Has a basic knowledge of modern programming languages and paradigms. Knows fundamental principles and structures to represent algorithm in the form of flowcharts. Knows the basic algorithms for searching, aggregation and sorting of the data.
PEU_W02	Knows the syntax, semantics, programming constructs and concepts specific to structured and procedural programming in C or C++. Understands concepts of iteration, recursion, memory organization, pointer arithmetic, dynamic resource allocation and release. Has the knowledge of the selected dynamic and complex data structures.
PEU_W03	Has the knowledge of modern software tools, information technologies and office software packages supporting the work of the programmer.
relating to skills:	
PEU_U01	Can represent an algorithm in the flowchart form. Can construct a solution for simple programming tasks that require the use of several branches, loops or recursion.
PEU_U02	Can properly structure the program code and data in C/C++, in accordance with the principles of structured and procedural programming. Can define and invoke functions, choose the way of passing the input and output parameters. Can define, initialize and process basic data representations: arrays, strings, structures and their combinations.
PEU_U03	Can appropriately use pointers and dynamic memory management, including proper allocation / deallocation procedures. Is able to design and program a set of functions that hide implementation details for complex and dynamic data structures. Can program the data storage operations in non-volatile memory using file-streams.
PEU_U04	Can use the integrated development environment to configure, edit, and test single-threaded console applications.
PEU_U05	Can effectively use business-office packages for preparing technical documentation, using spreadsheets for automating engineering calculations. Can extend their functionalities by programming new functions and macros.

PROGRAMME CONTENT		
Lecture		**Number of hours**
Lec 1	The algorithms and methods for their representation. The dominant programming paradigms. The flowcharts. The stages and tools used during software development. Standards of programming languages. The overall structure, syntax and semantics of the program in C or C++. Examples of source code for simple console applications.	2
Lec 2	Computer data and their representations. Data types and ranges of values. Program variables, variable declaration and initialization. The visibility of identifiers. Storage classes. Predefined scalar types and user defined types (typedef). Logic, bitwise and arithmetic operators. Rules for calculation of algebraic expressions. The standard mathematical functions. Dealing with streams and basic input/output operations. Dialogue with the user in text mode. Formatted input and output using standard libraries <stdio.h> <iostream>.	2
Lec 3	Basic programming instructions: assignment, conditional selection and choice. Controlling the flow of the algorithm, folding and nesting conditional instructions.	2

	Examples of algorithms that process small amounts of data (without using a loop). The concept of iterations in the program. The types of loops: while, do-while, for. Terms of completion and nesting the loops. Instructions to break or continue the loop. Simple iterative algorithms: counting, searching the minimum or maximum, summing up the data values retrieved from the stream.	
Lec 4	Arrays in C/C++. Array declaring, defining and indexing. Processing array data using a loop. One-dimensional and multi-dimensional arrays.	2
Lec 5	Functions and procedures in programming languages. Declaring, defining and invoking the function. Parameter-less functions. Explicit passing of the data via the argument list or the return statement. Passing arguments by value and by reference. Default values for arguments. Overloaded functions. Inline functions. Recursion.	2
Lec 6	Computer memory addresses, pointers to variables and memory, pointers arithmetic in C/C++. The relationship between pointers and arrays. Working with arrays using the pointer notation i. Passing arguments to the function by address. Standard C functions which operate directly on computer memory <mem.h> (memset, memcpy, memcmp, memmove, etc.)	2
Lec 7	Array representation of strings in C/C++. Declaring, defining, and manipulating the strings. Standard C library <string.h> (strcpy, strcmp, strcat, strlen, etc.). Examples of user-defined functions for processing textual data.	2
Lec 8	Program specification, testing, error handling, code documentation. Midterm (forming) exam	2
Lec 9	Recursion and recursive algorithms. Binary search and sorting of the arrays.	2
Lec 10	Structural type - the concept of structures in C/C++. Definition, declaration and initialization of structural variables. Nesting of composite types (structures and arrays). An example of a simple in+memory database using the representation in the form of arrays of structures.	2
Lec 11	Support for external memory in the form of raw data files. Random access and text files. Procedural <stdio.h> and object-oriented <fstream> <stream> libraries for standard file operations. Input and output operations for the characters, strings and formatted data. Binary data - block files. Portability of the data representation between different operating systems. Standards for exchange data files between applications written in C/C ++ and popular office suites (editors, spreadsheets).	2
Lec 12	Dynamic memory allocation. Allocating and freeing the allocated memory (malloc, calloc, free, new and delete operators). Heap overflow and dynamic data corruption. Dynamic allocation and reallocation of arrays of a specified size.	2
Lec 13	The complex pointer data structures. The array of pointers to simple variables, array of pointers to arrays, dynamic array of pointers to dynamic strings. Pointers to functions. Standard qsort function.	2
Lec 14	Dynamic and recursive data structures: the pointer-driven list, stack, queue, priority queue, binary tree, and their properties.	2
Lec 15	Utilizing the integrated office suites (editors, , spreadsheets, databases) in engineer work. Advanced processing capabilities of technical text documents and data in spreadsheets, through programming of new functions and macros.	2
	Total hours	**30**

Laboratory		Number of hours
Lab 1	Overview of the program and the organization of the laboratory classes. Workplace training in health and safety. Writing algorithms using flowcharts language. Setting up development environment (e.g. Windows/ Visual Studio or Linux/Emacs/gcc). An example of a console program using simple variables, assignment statements, and console input output operations. Editing, compiling, running and debugging the program. Guidelines for the use of integrated office packages for creating the technical documentation and reports on the implementation of laboratory tasks.	3
Lab 2	Representation of standard data types in C. Appropriate selection of the data type for variables. Data representation constraints. The dialogue with the user using standard printf and scanf functions. Formatting data (construction of format strings containing different control sequences). Calculating mathematical and boolean expressions in C/C++.	3
Lab 3	Exercises with the creation of example programs illustrating the use of basic C/C++ constructs and concepts: assignment, conditional branching (if, if-else), selection (switch, case, break, default). Nesting branching instructions.	3
Lab 4	The concept of iterations. The role and selection of the control variables for the loop. Loop breaking constructs (while, do-while, for). The equivalence of the loop.	3
Lab 5	Continuation of exercises with the creation of programs that illustrate the use of the user loop. Standard iterative algorithms: counting, summing, searching the maximum and minimum, calculation of the mathematical series.	3
Lab 6	Structured and procedural programming. Sub-division of tasks into functions, the concept of program menu. Visibility range and overriding the identifiers. Exercises with creating user-defined functions. Parameterless functions. Local variables. Passing parameters through global variables. The functions with explicit argument list. Passing arguments by value, reference and address.	3
Lab 7	Exercises with the creation of programs that illustrate the use of the array data representation. Processing arrays using a loop. Basic array processing algorithms (filling, comparing items, search, move, delete, add items).	3
Lab 8	Dynamic arrays (array with a counter of used items). Selected algorithms for processing arrays: linear and binary search, bubble sort and insertion sort. Parameterization of algorithms. Appropriate selection of the method for passing input/output parameters between the functions.	3
Lab 9	Text processing functions. Code analysis of the standard functions <string.h> library. User-defined functions for character string processing. Dynamic allocation and reallocation of memory. One-dimensional arrays of variable size. Pointer arithmetic and pointer casting. Exercise with accessing the memory through pointers.	3
Lab 10	Exercises with the creation of programs illustrating the processing of textual data, represented as an array of characters. Accessing the variables using pointers. Programs that use dynamic allocation and re-allocation of one-dimensional arrays. Debugging and testing the correctness of the programs.	3
Lab 11	Implementing simple in-memory database using representation in the form of an array of structures (or array of pointers to dynamic structures). Extending the functionalities of database program: adding archiving operations in the external memory (in the form of text or binary files).	3
Lab 12	The structural decomposition of large programs and complex data representation. Discussion and practice the representation of simple in-memory database (using an array of structures). User defined data type, enumeration. Encoding data using the dictionary.	3

Lab 13	Exercises with data storage in external memory using file streams. Text and binary representation of numerical data. Error detection during file stream input / output operations. Controlling the location of the file position indicator. Basic algorithms for sequential processing of text and raw binary files. Export/import of numeric and text data into popular office spreadsheet program.	3
Lab 14	User-defined implementation of selected dynamic data structure: the linked list, queue, priority queue or a tree. Exercises with creating programs using recursion.	3
Lab 15	Utilizing standard business office suites. Exercises with advanced formatting techniques of technical documents and performing engineering calculations using spreadsheets. Automation of work by programming new functions and macros.	3
	Total hours	**45**

TEACHING TOOLS USED
N1. Traditional lectures using multimedia projector N2. Individual work - self-implementation of appointed laboratory programs N3. Program code inspections carried out by the laboratory instructor N4. Individual work - self-study and preparation for lecture tests

EVALUATION OF SUBJECT LEARNING OUTCOMES ACHIEVEMENT

Evaluation (F – forming (during semester), P – concluding (at semester end)	Learning outcomes code	Way of evaluating learning outcomes achievement
F1	PEU_W01 – PEU_W03	Written test during the lecture. In the case of an additional test in the middle of the semester, the assessment F1 is a weighted sum of (1/3*F3 + 2/3*F4), where: F3 – evaluation of the midterm test F4 – evaluation of the final lecture test
F2	PEU_U01 – PEU_U05	Assessment of the reports documenting progress of laboratory exercises. Code inspection of the programs created by student, carried out by laboratory instructor.
P = 0.4*F1 + 0.6*F2, all partial evaluations must be positive		

PRIMARY AND SECONDARY LITERATURE

PRIMARY LITERATURE:

[1] Brian Kernighan, Dennis Ritchie, The C Programming Language, 1988
[2] Greg Perry, Dean Miller, C Programming Absolute Beginner's Guide, 3rd Edition, 2013
[3] Bjarne Stroustrup, The C++ programming language, 4th ed., 2013
[4] Stanley Lippman, Josée Lajoie, C++ primer, 5th ed., 2013,

SECONDARY LITERATURE:

[1] Niklaus Wirth, Algorithms + Data Structures = Programs, 1976
[2] Robert Sedgewick, Algorithms in C, 3rd Edition, 2001
[3] K.N. King, C Programming: A Modern Approach, 1996
[4] Dan Gookin, C for Dummies, Volume 1, 1994
[5] Alex Allain, Jumping into C++2013

SUBJECT SUPERVISOR (NAME AND SURNAME, E-MAIL ADDRESS)

Dr inż. Marek Piasecki, marek.piasecki@pwr.edu.pl