SYLLABUS

Name: ***Computer programming (MakAu>SI2CP19)***
Name in Polish:
Name in English: ***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:

ZAL

Language:

English

Course homepage:

https://platforma2.polsl.pl/rau2/course/view.php?id=89

Short description:

CP2 is the second of three parts of the computer programming course. The participant learns the object-oriented C++ language, assuming that he/she knows the basics of non-object-oriented programming in C/C++.

Description:

ECTS: 3
Total workload: 75 hours (40h contact hours, 35h students' own work hours)
Forms of contact hours:
Lecture: 15h
Laboratories: 15h
Other (e.g. test and reports revision and discussion): 10h
Students' own work: preparation for laboratories, revision of lecture slides, solving quizes, preparation for test

CP2 is the second of three parts of the computer programming course. The entire course provides the knowledge necessary to understand, design and write computer programs, especially in C and C++. The CP2 course covers the following topics: introduction to C++, reasons for using C++, non-objective C++ elements, language standards, object oriented programming (OOP) - general idea; classes, objects; constructors, destructors, operators, overloading; inheritance, derived classes, multiple inheritance; virtual methods; templates, exceptions; STL library. The CP2 course consists of a lecture and laboratory exercises. During the lecture, subsequent language elements are discussed and code examples are presented. Students receive all shown slides as PDF documents. Laboratory classes concern selected elements of the language that were discussed during the lecture. They begin with a short introduction - elements the lecture. Then a practical programming program is discussed, which should be solved using the discussed methods. Students individually solve the problem with the help of the teacher. They receive points for the problem solved during the class. Additionally, students write two individual programming projects in the form of homework. The first should be written in C or C++, the second should obligatory contain C++ elements (classes etc.). At the last lecture, a short quiz is conducted to test general knowledge of C++. This part of the course requires a good understanding of C programming. General math and logical skills are also required.
Detailed topics: The idea of object oriented programming, advanteges of OOP approach; non-object oriented extensions of C available in C++; cin, cout, namespaces, the namespace std; const values; references, functions accepting and returning references; inline functions; default arguments in C++ functions, overloading functions, specifying a default value for pointers. Basic input / output in C++: cin, cout, cerr, clog, using the std namespace, overloaded operators << and >>, manipulators, setw, setprecision, file input and output (details will be discussed later). Programming paradigms, the idea of object oriented programming; classes and objects; a structural and object oriented example – the class person with data members and methods; access rules (private, public, protected); default access rules in struct and class; hermetization and encapsulation; accessing class members; defining class methods – inside and outside of the class; the scope operator and qualifying names; creating objects, calling methods. Constructors; calling a constructor, passing parameters to constructors, temporary objects, defining constructors, initialization lists, classes and objects in other languages. Using the scope resolution operator ::. Static class members, static methods; friend declaration; friend methods, classes and standalone functions. Operators: overloading, calling. Operators defined as class methods and standalone functions, the number of required arguments, accessing operands from the method implementing an overloaded operator. The pre- and postincrement operator, the pre- and postdecrement operator. Order of evaluation, priorities. The assignment operator. Stream operators. Function call operator; conversion operator. Inheritance – introduction; accessibility of base class members, access modes, accessibility of base class members and methods in derived classes; constructors of derived classes; passing arguments to the constructor of the base class; order of calling of constructors and destructors; automatic calls of destructors of base classes, examples. Inheritance - continued; examples of converstion derived -> base; virtual methods, the VMT table, early and late binding; virtual destructors; copy constructor of derived classes, passing parameters, the initialization list; default constructors; move constructors, move assignment, copy assignment; generating default methods (=default, =delete); static methods and virtuality; pure virtual methods, abstract classes. RTTI - Run-Time Type Information - idea, the typeid operator, checking and comparing types. Type casting: traditional, C++ cast operators: static_cast, dynamic_cast, reinterpret_cast, const_cast - introduction. Templates - class templates, function templates.

Bibliography:

1. Bjarne Stroustrup: The C++ Programming Language
2. Bjarne Stroustrup: Język C++ Kompendium wiedzy
3. Bjarne Stroustrup: A Tour of C++
4. Stanley Lippman, Josée Lajoie, Barbara Moo: C++ Primer
5. Bjarne Stroustrup: Principles and Practice Using C++
6. Online books and tutorials, https://www.learncpp.com

Learning outcomes:

Knowledge: the student knows and understands
(K1A_W11) detailed issues of algorithms, numerical methods, programming in selected languages, programming engineering calculations,
Skills: the student is able to
(K1A_U21) implement algorithms using the programming language they have learned, analyze algorithms
(K1A_U22) create simple applications operating under the control of various hardware and software environments

Assessment methods and assessment criteria:

Assessment methods and assessment criteria:

(5 - performing an exercise, 6 - reporting on a laboratory exercise, 11 - participating in classes) Performing exercises related to the lecture topics (checking the ability to perform a task);
(7 - performing a project, 8 - report on the project) Performing a programming project (checking the ability to independently perform a project and implement a given task - developing an application that works in a selected environment; skills in selecting the right techniques and tools to solve the task and test the solution, preparing documentation);
(4 - final test) Lecture final test (open and closed test, single and multiple choice).
For laboratory exercises, you can receive up to 10 points in total, for the individual programming task up to 12 points, for the lecture test up to 8 points. The condition for obtaining a pass is to receive at least 50% of the points in each category. After meeting this condition, the final grade of 3.0 to 5.0 is determined depending on the sum of points, with ranges: [15, 18), [18, 21), [21,24), [24, 27), [27, 30] for grades: 3.0, 3.5, 4.0, 4.5, 5.0, respectively.

The syllabus is valid from the summer semester of the academic year 2025/26, and its content is not subject to change 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)	3	2020/2021-L	