# SYLLABUS / FIŞA DISCIPLINEI

## 1. Information on the study programme

| | |
|---|---|
| 1.1. Institution | West University of Timişoara |
| 1.2. Faculty | Mathematics and Computer Science |
| 1.3. Department | Computer Science |
| 1.4. Study program field | Computer Science |
| 1.5. Study cycle | Undergraduate |
| 1.6. Study programme / Qualification | Computer Science : *Database administration / Administrator baze de date - 252101; Computer network administration / Administrator de retea de calculatoare - 252301; Analyst / Analist - 251201; Research assistant in computer science / Asistent de cercetare în informatica - 214918; Teacher in secondary schools / Profesor în învatamântul gimnazial - 233002; Programmer / Programator - 251202; Software systems designers / Proiectant sisteme informatice - 251101* |

## 2. Information on the course

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2.1. Course title | | | Programming II | | | | |
| 2.2. Lecture instructor | | | Daniel Pop, Cosmin Bonchiş | | | | |
| 2.3. Seminar / laboratory instructor | | | Adrian Florin Spătaru, Florin Rosu, | | | | |
| 2.4. Study year | 1 | 2.5. Semester | 2 | 2.6. Examination type | E | 2.7. Course type | DI |

## 3. Estimated study time (number of hours per semester)

| | | | | | |
|---|---|---|---|---|---|
| 3.1. Attendance hours per week | 4 | out of which: 3.2 lecture | 2 | 3.3. seminar/laboratory | 2 |
| 3.4. Attendance hours per semester | 56 | out of which: 3.5 lecture | 28 | 3.6. seminar/laboratory | 28 |

| **Distribution of the allocated amount of time*** | **hours** |
|---|---|
| Study of literature, course handbook and personal notes | 35 |
| Supplementary documentation at library or using electronic repositories | 15 |
| Preparing for laboratories, homework, reports etc | 40 |
| Exams | 6 |
| Tutoring | 8 |

| | |
|---|---|
| 3.7. Total number of hours of | 104 |

| | |
|---|---|
| individual study | |
| 3.8. Total number of hours per semester | 160 |
| 3.9. Number of credits (ECTS) | 6 |

**4. Prerequisites (if it is the case)**

| | |
|---|---|
| 4.1. curriculum | Programming I, Algorithms and Data structures I |
| 4.2. competences | Problem solving abilities, Proficiency in English |

**5. Requirements (if it is the case)**

| | |
|---|---|
| 5.1. for the lecture | Room equipped with beamer and whiteboard<br>Online: Google Classroom Code: odjjuhr<br>Google Meet / Cisco Webex / Microsoft Teams |
| 5.2. for the seminar, laboratory | Room equipped with computers running one of the following IDEs: Code Blocks, MS Visual Studio (Academic licence), Eclipse (with C++ plugin), IntelliJ IDEA (with C++ plugin)<br>Online: Google Classroom Code: odjjuhr<br>Google Meet / Cisco Webex / Microsoft Teams |

**6. Specific acquired competences**

| | |
|---|---|
| Professional skills | • Good knowledge of procedural and object-oriented concepts<br>• Introduction to generic programming<br>• Ability to implement small-sized procedural and object-oriented projects in C/C++<br>• Usage of an Integrated Development Environment |
| Transversal skills | • Ability to build complex systems based on elementary building blocks<br>• Develop an analytical spirit and curiosity about how computer software works<br>• Conceptual modelling, i.e. ability to represent real-life problems using abstract models |

**7. Course objectives**

| | |
|---|---|
| 7.1. General objective | Assimilate and ability to operate with procedural and object-oriented paradigm in C/C++; develop an object-oriented mindset |
| 7.2. Specific objectives | *Knowledge wise objectives (KO):* (1) Good knowledge of procedural and object-oriented paradigms (2) Basic knowledge of generic programming concepts (3) Basic conceptual modelling and object-oriented |

| | |
|---|---|
| | design; (4) Code the concepts of procedural programming in C programming language; (5) Represent OOP and generic programming concepts in C++ programming language<br><br>*Ability wise objectives (AO):* (1) Ability to design simple problems using OOP concepts; (2) Ability to implement and test simple problems in C and C++ programming languages; (3) Ability to debug programmes.<br><br>*Skills wise objectives (SO):* (1) Root cause analysis; (2) Generalization and conceptualization of real-life problems |

**8. Content** *

| **8.1. Lecture** | **Teaching methods** | **Remarks, details** |
|---|---|---|
| C1 (2h) Introduction to C/C++ programming languages. Short history. C/C++ program structure. Building stages. Operators (incl. bitwise ops) | Lecture, discussion, active student participation | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 2] |
| C2 (2h) Data types. Declarations. | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 6] |
| C3 (2h) User-defined functions and macros | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 12] |
| C4 (2h) Pointers. Arrays. References | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 7] |
| C5 (2h) Structures. Unions. Bitfields. Enumerations | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 8] |
| C6 – C7 (4h) Object oriented programming concepts in C++. Classes: access control. Constructors. Destructor. Self-reference. Modifiers. Scopes. | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 16] |
| C8 (2h) Objects: object lifecycle. | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 16] |
| C9 (2h) Inheritance. Derived classes. Accessing base class | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 20] |

| | | |
|---|---|---|
| members. Constructors. Destructor. Virtual functions. | | |
| C10 (2h) Inheritance. Abstract classes. Polymorphism. Class hierarchies. Multiple inheritance | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 16] |
| C11 (2h) Exception handling. Definition. Throw-try-catch mechanism. Exception | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 13] |
| C12 (2h) Generic programming. Introduction. Template classes. Template functions. | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 23] |
| C13 (2h) Generic programming. Specializations. Inheritance. Covariance. Contravariance | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 25] |
| C14 (2h) C++ Standard Library. Introduction. Containers. Algorithms. Iterators. Strings. Streams. Numerics | Idem | Bjarne Stroustrup – The C++ Programming Language 4th Edition, Addison Wesley, 2013 [Chapter 30] |
| **Recommended bibliography / Bibliografie**<br>[1] Bjarne Stroustrup – The C++ Programming Language 4th Edition. Addison Wesley, 2013.<br>[2] Brett McLaughlin, Gary Pollice, David West – Head First Object-Oriented Analysis and Design, O'Reilly, 2006 | | |
| **8.2. Seminar, laboratory** | **Teaching methods** | |
| L1 (2h) Basic concepts: data types, I/O functions, conditional instructions. | Students will be organized in two or more breakout rooms;<br>Students will take turns in sharing screens while resolving the challenges;<br>Lab instructors will offer guidance and explanations;<br>Home assignments will be used to complete additional exercises | Labs will be conducted online using Stepik platform<br>https://stepik.org/course/52108/syllabus |

| | | |
|---|---|---|
| L2 (2h) User-defined functions. Macros | Idem | Idem |
| L3 (2h) Arrays and character strings | Idem | Idem |
| L4 (2h) Pointers | Idem | Idem |
| L5 (2h) Test for C Programming Language | Test on Stepik | Test on Stepik |
| L6 (2h) Default argument values. I/O streams. Reference type. Memory allocation in C++ | Hands-on<br>Discussions, problem analysis<br>Homework | https://stepik.org/course/52108/syllabus |
| L7 (2h) Class declarations. Constructors and destructor. Const modifier | Idem | Idem |
| L8 (2h) Modifiers: static. Nested classes and objects | Idem | Idem |
| L9 (2h) Class relationships | Idem | Idem |
| L10 (2h) Inheritance and polymorphism | Idem | Idem |
| L11 (2h) Exception handling | Idem | Idem |
| L12 (2h) Generic programming. Template classes and functions | Idem | Idem |
| L13 (2h) Generic programming. Standard Template Library (STL) | Idem | Idem |
| L14 (2h) Test for C Programming Language | Test on Stepik | Test on Stepik |
| **Recommended bibliography**<br>[1] https://bitbucket.org/danielpop/programming-ii-labs/src/master/<br>[2] Laboratories on Stepik platform: https://stepik.org/course/52108/syllabus<br>[3] Bjarne Stroustrup – The C++ Programming Language 4th Edition. Addison Wesley, 2013 | | |

**9. Correlations between the content of the course and the requirements of the professional field and relevant employers**

| |
|---|
| It is nearly impossible to develop applications nowadays that are not written in an object-oriented language. Object-oriented modelling and implementation is the de-facto approach used to implement complex systems across multiple businesses, such as financial, commercial, industrial or online commerce. The local, national and international workforce market is continuously looking for highly-skilled personnel to develop complex systems using object-oriented languages and environments. |

**10. Evaluation***

| Activity | 10.1. Assessment criteria ** | 10.2. Assessment methods *** | 10.3. Weight in the final mark |
|---|---|---|---|
| 10.4. Lecture | • *(KO*1) Good knowledge of procedural and object-oriented paradigms<br>• (KO2) Basic knowledge of generic programming concepts<br>• (KO3) Basic conceptual modelling and object-oriented design;<br>• (KO4) Code the concepts of procedural programming in C programming language;<br>• (KO5) Represent OOP and generic programming concepts in C++ programming language | Written test<br><br>Short quizzes at each lecture | 80% of final lecture grade, which is 50% of final grade<br><br>20% of final lecture grade, which is 50% of final grade |
| 10.5. Seminar / laboratory | • (AO1) Ability to design simple problems using OOP concepts;<br>• (AO2) Ability to implement and test simple problems in C++ programming language;<br>• (AO3) Ability to debug running programmes<br>• (SO2) Generalization and conceptualization of real-life problems | Practical test 1 – Lab 5 (covers C programming language) | 33% of final practical grade, which is 50% of final grade |
| | | Practical test 2 – Lab 14 (covers C++) | 66% of final practical grade, which is 50% of final grade |
| 10.6. Minimal needed performance for passing | | | |
| Minimal knowledge for passing this subject: | | | |

- Good knowledge of basic concepts of procedural and object-oriented paradigms in C/C++ languages
- Ability to model a simple problem using object-oriented concepts
- Ability to understand and model simple problems using template classes and functions
- Ability to write a simple program (up to 10 related classes) in C++ programming languages, to compile it and run it successfully
- Ability to use basic collections and algorithms from STL (Standard Template Library)

For students in 1st year of study: The final grade is computed as the average of grades obtained for components described in 10.4 and 10.5. The exam is passed if each individual grade obtained at components 10.4 and 10.5 (i.e. both lecture and lab evaluations) are greater or equal to 5. The practical work evaluation (labs) grade is computed as the average of the two tests scheduled in Lab 6 and 14, respectively. **Only one** of the two practical tests can be retaken in the examination session(s) (B1-B3). If both tests are failed the student needs to re-take the subject next year! The lecture assessment is composed of a final online test weighted with the results obtained at each lecture quiz. The student needs to re-take only the failed component (course or one of the lab tests, respectively) during the re-examination sessions, unless the student wishes to re-take both evaluations. If the student wishes to increase a passing grade, both components (10.4 and 10.5) need to be re-evaluated.

For students in 2nd and 3rd year of study who need to retake the exam or are re-enrolled on this subject: **you need to re-take both tests** (lecture + lab evaluation). You have to come in Lab 6 / 14 when tests are given. At exam you can only take one of the 2 practical tests (same as for students in year 1). **Grades obtained years ago are not considered!**

Final remark: All students all welcome to tutoring meetings as scheduled by the department.

| Date | Signature (lecture instructor) | Signature (seminar instructor) |
|---|---|---|
| 23/09/2020 | Conf. Dr. Daniel Pop | As. Drd. Adrian Florin Spătaru |

Signature (director of the department)