Course unit title:	Introduction to Programming		
Course unit code:	AMDM182		
Type of course unit:	Required		
Level of course unit:	Bachelor (1st cycle)		
Year of study:	1		
Semester when the unit is delivered:	1 (Fall)		
Number of ECTS credits allocated :	6		
Learning outcomes of the course unit:	1. Familiarize with main elements of a computer system, a computer program and computer applications 2. Recognise the goals, capabilities and benefits of structured programming and the basis of algorithmic thought. 3. Examine written programs and identify their function and underlying algorithmic logic. 4. Demonstrate the ability to express elementary algorithms in the syntax of an imperative programming language by using a programming environment such as Alice + Java or Scratch. 5. Demonstrate the ability to apply correct operations and form the necessary statements. 6. Analyse simple problems, construct algorithms to programmatically solve them, and formulate corresponding programs using selective, iterative and sequential statements. 7. Illustrate the ability to define and use arrays programmatically. 8. Recognize and illustrate the predefined functions, and user-defined functions prototypes, definitions, and calls.		
Mode of delivery:	Face-to-face		
Prerequisites:	None	Co-requisites:	None
Recommended optional program components:	None		
Course contents:	• **Introduction to Computer Programming:** Computer Systems: Hardware, Networks, Computer organization, Computer memory, Computer software, Running a program, High-level languages, Low-level languages, Compilers. Programming and Problem-Solving: Algorithms, Logic Diagrams, Pseudocode, Flowcharts, Program design, Problem solving phase, Implementation phase, Programming guidance, Programming steps, Program creation, Object Oriented Programming (OOP), OOP characteristics, Software life cycle. • **Programming Basics:** A sample Java program. Explanation of code. Program layout (include directives, main function, variables, comments). Running a Java program. Testing and Debugging. Program errors. Variables and Assignments: Identifiers, Keywords, Declaring variables, Assignment statements, Initializing variables. Programming software: e.g., Alice or Scratch. Get familiar with the environment and be able to execute simple statements and commands. • **Conditional Statements:** Flow of Control. Branch. Designing the branch. Implementing the branch. if-else syntax. Boolean expressions. Relational operators. if-else flow of control. Logical operators. Compound statements. Program Style: Indenting, Comments, Constants. Using boolean expressions. Evaluating boolean expressions. Truth tables. Order of precedence. Precedence rules. Short-Circuit evaluation. Type bool and Type int. bool return values. Multiway branches. Nested statements. Nested if-else statements. Multi-way if-else statements. The switch-statement: syntax, the controlling statement, the		

	break statement, the default statement, Switch-statements and menus. Blocks with local variables. Statement blocks. Scope rule for nested blocks. • **Repetitive Statements:** Loop statements. while-loop syntax, operation and flow of control. do-while loop. Infinite loops. Prefix & Postfix Increment/Decrement Operators. The for-statement. for/while loop comparison. Which loop to use. The break-statement. Designing Loops. Ending a loop: List headed by a size, Ask before iterating, List ended with a sentinel value. Running out of input. General methods to control loops: Count controlled loops, Exit on flag condition, Exit on flag caution. Nested loops. Debugging loops. Fixing Off by one errors. Fixing infinite loops. Tracing variables. Loop testing guidelines. • **Program Modularity:** Top-down design. Predefined functions. Function calls. Function call syntax. Function Libraries. Programmer-defined functions. Function declaration. Function definition. The return statement. The function call. Alternate declarations. Order of arguments. Function definition syntax. Placing definitions. Procedural abstraction. Information hiding. Formal parameter names. Local variables. Global constants and variables. void-functions, Call-by-reference parameters, Choosing parameter types. • **Composite Data types:** Declaring an array. The array variables. Array variable types. Indexed variable assignment. Loops and arrays. Constants and arrays. Array declaration syntax. Arrays and memory. Array index out of range. Out of range problems. Initializing arrays. Default values. Uninitialized arrays. Programming with arrays. Searching arrays. The search function. Sorting an array. The selection sort algorithm. Sort algorithm development. • **Laboratory Work:** The role of a programming language as a tool for solving simple and complex problems is emphasised through practical work carried out.
Recommended and/or required reading:	
Textbooks:	Dann, W.P. and Cooper, S. and Pausch, R.,Learning to Program with Alice, Pearson Education, 2011
References:	Paul M. Mullins and Michael Conlon. 2008. Engaging students in programming fundamentals using alice 2.0. In *Proceedings of the 9th ACM SIGITE conference on Information technology education* (SIGITE '08). ACM, New York, NY, USA, 81-88. Paul Mullins, Deborah Whitfield, and Michael Conlon. 2009. Using Alice 2.0 as a first language. *J. Comput. Sci. Coll.* 24, 3 (January 2009), 136-143. Maloney, John; Hernández, Andrés; Rusk, Natalie; Eastmond, Evelyn; Brennan, Karen; Millner, Amon; Rosenbaum, Eric; Silver, Jay; Silverman, Brian; Kafai, Yasmin (November 2009). "Scratch: Programming for All". Communications of the ACM 52 (11): 60–67.
Planned learning activities and teaching methods:	Students are taught the course through lectures by means of computer presentations. Laboratory work consists of practical problems aiming to help students understand and illustrate the programming concepts taught at lectures. Homework requires students to solve programmatically simple problems. Lecture/Laboratory notes and presentations are available through the web for students to use in combination with the textbooks.
Assessment methods and criteria:	• Lab Assignments: 20% • Tests: 20% • Final Exam: 60%
Language of instruction:	English
Work placement(s):	No