Course Title	Programming Principles I				
Course Code	ACSC182				
Course Type	Compulsory				
Level	BSc (Level 1)				
Year / Semester	1st (Fall)				
Teacher's Name	Chrysostomos Chrysostomou				
ECTS	5	Lectures / week	2	Laboratories/week	2
Course Purpose	This course aims to introduce students to programming principles using the C++ programming language with emphasis on the algorithmic design, and program development through information representation, assignments and operations, conditional and repetitive statements, composite data type (arrays), and modularity (functions). The role of the C++ programming language as a tool for solving simple and complex mathematical and engineering problems is emphasized through practical work carried out.				
Learning Outcomes	By the end of the course, the students are expected to: 1. recognize the goals, capabilities and benefits of structured programming and the basis of algorithmic thought; 2. examine written programs and identify their function and underlying algorithmic logic; 3. demonstrate the ability to express algorithms in the syntax of an imperative programming language (C++); 4. choose the appropriate data types, apply the correct operations, and form the necessary statements; 5. analyze simple to complex problems, construct algorithms to programmatically solve them, and formulate corresponding programs using selective, iterative and sequential statements; 6. illustrate the ability to define and use one- and multi-dimensional arrays programmatically; 7. recognize and illustrate the predefined functions, and user-defined functions prototypes, definitions, and calls.				
Prerequisites	None		Co-requisites	None	
Course Content	• **Introduction to Computer Programming:** Computer Systems: Hardware, Networks, Computer organization, Computer memory, Computer software, Running a program, High-level languages, Low-level languages, Compilers, Compiling and running a C++ program, Linkers. 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 C++ program. Explanation of code. Program layout (include directives, main function, variables, comments). Running a C++ program. Testing and Debugging. Program errors. Variables and Assignments: Identifiers, Keywords, Declaring variables, Assignment statements, Initializing variables. Input and Output: Output using cout, Include directives, Escape sequences, Formatting real numbers, Showing decimal places, Basic cout manipulators, Input using cin, Reading data from cin, Designing input and output. Data Types and Expressions: Writing integer constants, Writing double constants, Other number types, Integer types, Floating point types, Type char, char constants, Reading character data, Type string, Type bool, Type compatibilities (int - double, char - int, bool - int), Arithmetic, Results of operators, Division of doubles, Division of integers, Integer remainders, Arithmetic expressions, Operator shorthand. • **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 error. 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, Mixed parameter lists. 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. Arrays in functions. Arrays as function arguments. Array parameter declaration. Function calls with arrays. Function call details. Array formal parameters. Array argument

<table>
<tr><td></td><td>details. Array parameter considerations. Programming with arrays. Partially filled arrays. Searching arrays. The search function. Sorting an array. The selection sort algorithm. Sort algorithm development. Multi-dimensional arrays. Multi-dimensional parameters.
• Laboratory Work: The role of the C++ programming language as a tool for solving simple and complex mathematical and engineering problems is emphasised through practical work carried out.</td></tr>
<tr><td>Teaching Methodology</td><td>Students are taught the course through lectures by means of computer presentations. Lectures are supplemented with laboratory work and assignments that consist of simple and complex mathematical and engineering problems aiming to help students develop practical skills by illustrating the programming concepts taught at lectures.
Lecture/Laboratory notes and presentations are available for students to use in combination with the textbooks, through the university’s e-learning platform.</td></tr>
<tr><td>Bibliography</td><td>Textbook:
• Walter Savitch, Problem Solving with C++, Pearson, 10th Edition, 2018
Reference:
• Paul J. Deitel, Harvey M. Deitel, C++ How to Program, Pearson, 10th Edition, 2017</td></tr>
<tr><td>Assessment</td><td>The assessment of the course includes two lab tests, one written test and a final written exam with programmatically problem-solving questions. Laboratory work consists of practical problems aiming to help students understand and illustrate the programming concepts taught at lectures, and assignments requiring students to solve programmatically simple and complex mathematical and engineering problems.
The weights for each assessment component are:
<table>
<tr><th colspan="2">Assessment Weights:</th><th>Partial</th><th>Total</th></tr>
<tr><td rowspan="5">Continuous Assessment</td><td>One Written Test</td><td>40%</td><td rowspan="5"></td></tr>
<tr><td>Two Lab Tests</td><td>25%</td></tr>
<tr><td>Two Lab Assignments</td><td>20%</td></tr>
<tr><td>Lab exercises</td><td>15%</td></tr>
<tr><td>Continuous Assessment:</td><td>100%</td><td>40%</td></tr>
<tr><td colspan="3">Final Exam</td><td>60%</td></tr>
</table></td></tr>
<tr><td>Language</td><td>English</td></tr>
</table>